Data Augmentation for Attack Detection on IoT Telehealth Systems

by

Zaid Ali Khan
B.S., NED University of Engineering and Technology, 2013

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

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Data Augmentation for Attack Detection on IoT Telehealth Systems

by

Zaid Ali Khan
B.S., NED University of Engineering and Technology, 2013

Supervisory Committee

---

Dr. Fayez Gebali, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Watheq El-Kharashi, Co-Supervisor
(Department of Electrical and Computer Engineering)

**Supervisory Committee**

---

Dr. Fayez Gebali, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Watheq El-Kharashi, Co-Supervisor
(Department of Electrical and Computer Engineering)

## ABSTRACT

Telehealth is an online health care system that is extensively used in the current pandemic situation. Our proposed technique is considered a fog computing-based attack detection architecture to protect IoT Telehealth Networks. As for IoT Telehealth Networks, the sensor/actuator edge devices are considered the weakest link in the IoT system and are obvious targets of attacks such as botnet attacks. In this thesis, we introduce a novel framework that employs several machine learning and data analysis techniques to detect those attacks. We evaluate the effectiveness of the proposed framework using two publicly available datasets from real-world scenarios. These datasets contain a variety of attacks with different characteristics. The robustness of the proposed framework and its ability, to detect and distinguish between the existing IoT attacks that are tested by combining the two datasets for cross-evaluation. This combination is based on a novel technique for generating supplementary data instances, which employs GAN (generative adversarial networks) for data augmentation and to ensure that the number of samples and features are balanced.

# Contents

**Supervisory Committee** **ii**

**Abstract** **iii**

**Contents** **iv**

**List of Tables** **vi**

**List of Figures** **vii**

**Acronyms** **viii**

**Acknowledgements** **x**

**Dedication** **xi**

**1 Introduction** **1**

1.1 Problem Statement . . . . . 1

1.2 Goals . . . . . 2

1.3 Thesis Organization . . . . . 2

**2 Related Work** **4**

**3 Background** **6**

3.1 Telehealth model: Edge & Fog Computing-based system . . . . . 6

3.2 Edge Computing (IoT edge devices) . . . . . 8

3.3 Fog Computing (Internet Cloud) . . . . . 8

3.4 IoT attacks . . . . . 9

3.4.1 IoT Ecosystem Characteristics . . . . . 9

3.4.2 Vulnerabilities in the IoT . . . . . 12

3.4.3 Security Aspects . . . . . 14

3.4.4 IoT Security attacks . . . 15
3.5 Machine Learning Methods . . . 16
3.6 Evaluation criteria . . . 17
3.7 Datasets . . . 18
3.7.1 ToN-IoT Telemetry dataset . . . 18
3.7.2 IoTID20 Dataset . . . 18
3.8 GAN (Generative Adversarial Networks) . . . 18

**4 Methodology** **20**
4.1 Framework . . . 20
4.2 Preparation and pre-processing . . . 21
4.2.1 Data Normalization . . . 24
4.3 Features Selection . . . 25

**5 Experimental setup & Results** **31**
5.1 Experiment 1: Ton-IoT dataset . . . 31
5.1.1 Benign & Malicious Classification . . . 31
5.1.2 Multi-Class Classification . . . 32
5.2 Experiment 2: IoTID20 dataset . . . 33
5.2.1 Benign & Malicious Classification . . . 33
5.2.2 Multi-Class Classification . . . 34
5.3 Experiment 3: Combined ToN-IoT & IoTID20 datasets . . . 35
5.3.1 Experiment 3a . . . 37
5.3.1.1 Benign & Malicious Classification . . . 37
5.3.1.2 Multi-Class classification . . . 37
5.3.2 Experiment 3b . . . 38
5.3.2.1 Benign & Malicious Classification . . . 38
5.3.2.2 Multi-Class Classification . . . 39

**6 Conclusion and Future Work** **41**

**Bibliography** **43**

# List of Tables

Table 4.1 ToN-IoT Binary Data . . . . . 23
Table 4.2 ToN-IoT Anomaly Data . . . . . 23
Table 4.3 IoTID20 Binary Data . . . . . 23
Table 4.4 IoTID20 Anomaly Data . . . . . 24
Table 4.5 ToN-IoT Feature Description [1] . . . . . 25
Table 4.6 IoTID20 features removed due to correlation . . . . . 26
Table 4.7 IoTID20 Deleted Features . . . . . 27
Table 4.8 IoTID20 Feature Description [2] [3] . . . . . 27

Table 5.1 Benign & Malicious Classification with ToN-IoT dataset . . . . 32
Table 5.2 ToN-IoT dataset Multi-Class Classification . . . . . 33
Table 5.3 IoTID20 dataset Benign and Malicious Classification . . . . . 34
Table 5.4 IoTID20 dataset Multi-Class Classification . . . . . 34
Table 5.5 Benign & Malicious attacks on DS for training and IoTID20 $DS_a$ dataset . . . . . 37
Table 5.6 Multi-class Classification with DS for training and IoTID20 $DS_a$ dataset . . . . . 38
Table 5.7 Benign and Malicious Classification with DS for training and ToN-IoT $DS_b$ dataset . . . . . 39
Table 5.8 Multi-class Classification with DS for training and ToN-IoT $DS_b$ dataset . . . . . 39

# List of Figures

Figure 3.1 Basic IoT telehealth system structure represents client-side [4]. 7
Figure 3.2 Basic IoT telehealth system structure represents server-side [4]. 8
Figure 3.3 IoT ecosystem characteristics. . . . . 10
Figure 3.4 Generative Adversarial Networks . . . . . 19

Figure 4.1 Proposed Attack Detection System Framework. . . . . 21
Figure 4.2 Applied normalization on ToN-IoT on first two features. . . . . 25

Figure 5.1 Representation of datasets based on Attacks and Normal samples [5] . . . . . 35
Figure 5.2 Combined dataset using GAN's simulated data by real datasets [5] 36
Figure 5.3 GAN generated samples of single feature from each dataset . . 36

# Acronyms

**A** Accuracy

**Ada** AdaBooting Classifier

**ARM** Advanced RISC Machines

**ASLR** Address Space Layout Randomization

**CSV** Comma Separated Values

**DDoS** Distributed Denial of Service

**DoS** Denial of Service

**DT** Decision Tree Classifier

**F** $F_1$-score

**FTP** File Transfer Protocol

**GAN** generative adversarial networks

**GB** Gradient Boosting Classifier

**HRoT** Hardware Based Root-of-trust

**HTTP** Hypertext Transfer Protocol

**IMAP** Internet Message Access Protocol

**IoT** Internet of things

**MIM** Man-in-the-middle attack

**MIPS** Million instructions per second

**MITM** Man-in-the-middle attack

**MMU** Memory Management Unit

**NB** Naive Bayes Classifier

**P** Precision

**POP3** Post Office Protocol 3

**R** Recall

**RF** Random Forest Classifier

**Scan** Scanning Attack

**SEIT** School of Engineering and Information technology

**SMTP** Simple Mail Transfer Protocol

**SQL** Structured Query Language

**SSH** Secure Socket Shell

**TCP** Transmission Control Protocol

**TLS** Transport Layer Security

**UDP** User Datagram Protocol

**UNSW** University of New South Wales

**WiFi** Wireless Fidelity

**XGB** XGBoost Classifier

**XSS** Cross Site Scripting

**ZB** ZettaByte

**ZigBee** Zonal Intercommunication Global-standard

**Z-Wave** A wireless communication protocol used primarily in smart home networks

# ACKNOWLEDGEMENTS

First of all, I would like to express my deep gratitude, thanks, and prayers to the Mighty Allah, the most gracious and the most merciful, who blessed us with the ability to think, taught us everything, and gave me the power to finish this work. I wish to highlight my sincere gratitude to Dr. Fayez Gebali as my supervisor whose support, guidance, expertise, flexibility, and encouragement contributed greatly to my graduate studies. His knowledge helped me to complete the thesis. His flexibility allowed me to confidently work remotely on my thesis even in this pandemic as we worked remotely all the time. The author acknowledges the support of the National Research Council (NRC) of Canada under the Collaborative R&D Initiative HQP Grant Application. Thankful to Dr. Elhadad for his contribution to the thesis. Finally, I am also thankful to the University of Victoria for providing me an opportunity to study here and also providing resources that enabled me to generate the results for my research.

# DEDICATION

*To my parents, for their continuous support, love, and prayers.*

*To my supervisor, Dr. Fayez Gebali for his constant support, guidance, and flexibility.*

# Chapter 1

# Introduction

Nowadays, IoT is used in many infrastructure systems such as communications, commerce, healthcare, manufacturing, smart homes, etc. [4], [6]. IoT systems are a natural target to cyber-attacks that target IoT servers or IoT edge devices [7]. Attacks on telehealth infrastructure IoT systems are particularly of importance given the present pandemic situation. Telehealth IoT edge devices are a natural target for cyberattacks since these devices are located in unsecured locations and have none or primitive operating systems. Furthermore, the devices have very limited energy and compute resources. These vulnerabilities pose a serious threat to IoT usage in telehealth and machine learning has become necessary for real-time attacks detection.

## 1.1 Problem Statement

IoT devices are targets for attackers as encryption and authentication are not easy to implement on these devices. The threat is always around to gain unauthorized and unsafe access by malicious attacks due to the vulnerabilities of these devices. This lack of security leads to a common and easy gateway that is being exposed to the attackers. Although researchers are working to make them robust and secure to tackle these attacks. To prevent IoT devices from these attacks, we need detection techniques that would classify intrusion attacks. Machine learning classification algorithms are one of the solutions for this problem that can detect these attacks using trained models.

## 1.2 Goals

Classification of IoT attacks using machine learning classifiers is a complex problem and it becomes more complicated when it comes to adapting the correct machine learning approach and relevant datasets. From previous studies, we can see that all approaches gave us different results. In short, there is no universal approach for feature extraction and classification because the result of each experiment changes with changes in the dataset or the way classification is performed via their evaluation approach.
The objective of the research is to create a hybrid cross-evaluation approach by training machine learning models on each and combining datasets to determine which classification approach gives us the best results using different feature extraction techniques. Multiple classifiers are used along with ensemble methods and their performances are measured when the best parameters are used. Recent studies have shown that different results are obtained when using different machine learning models. In this thesis, we aim to follow the below goals to solve this problem.

1. We used GANs to combine and augment two IoT attack datasets.
2. We employed feature engineering and preprocessing on the combined dataset.
3. We performed and compared classification of IoT security attacks using multiple machine learning classifiers

## 1.3 Thesis Organization

The remaining of the thesis is organized as follows.

**Chapter 2** Includes the detailed related work recently has been done in the relevant field and reviews that used the same datasets.

**Chapter 3** It is related to the theory about the problem with its vulnerabilities and datasets we used, knowledge about machine learning, IoT device attacks, and types.

**Chapter 4** Includes the approach used for the feature engineering along with the pre-processing which is described in detail. In addition, the proposed framework and its functionality.

**Chapter 5** The performance of Random forest, Decision tree, Logistic regress, K nearest neighbors, XGBoost, and AdaBoost classifiers on each and combined dataset. It shows the result tables and discussions about the iterations based on evaluation approaches used for the comparison of performance.

**Chapter 6** Concludes and summarizes the goal achievements from the experiments.

# Chapter 2

# Related Work

In this section, we discuss the recent work that has been done on IoT intrusion detection and classification.

Many researches were conducted using *KDD Cup99* and *NSL-KDD* data sets. These datasets are available over the internet. These datasets did not cover new variants of attacks [8]. *DEFCON-8* dataset is limited because of the low number of records which limits the experimental approach and may lead towards less accurate results when executing real-time network traffic scenarios like botnets.

A new dataset has been proposed *CICIDS2017*. This dataset is based on more diverse attacks such as DoS, DDoS, XSS, SQL Injection, Brute force, Data Infiltration, portscan, and Botnet [9]. The dataset has 80 flow-based features generated using CICFlowMeter [10] [11] and focuses on flow-identification packets and features. S. Ustebay et al. [12] used the same dataset *CICIDS2017* to create an anomaly detection system based on recursive feature elimination. The dataset was unbalanced so it was reduced by 95% for the experiments. The training and testing samples were split in between 80% for training and 20% for testing. The model is trained for binary classification and resulted a score of 91%. Furthermore, they eliminated flow identifier features while training the model.

*IoTID20* dataset was used in [13] and used to employ the test ability of different intrusion detection algorithms in IoT networks [2]. This dataset has diverse attacks samples extracted in terms of real-time scenarios. The binary and multi-class classification was performed using the dataset. The authors used different classification methods and reported the results using cross-validation. The decision tree performed well and gave the highest accuracy of 88%. They used the flow-identifiers in their training set. This has been so far the relative and recent dataset that we will use

in our work. We use another dataset called *Ton-IOT Telemetry dataset* [1]. It was developed by using sensors and smart devices and simulated multiple attack samples. Alsaedi et al. used different machine learning models on the *Ton-IOT telemetry dataset* after experimenting on each dataset along with the combined dataset. Their experiments included binary and multi-class classification. They split the dataset with the ratio of 80% and 20% for training and testing respectively. They used 4-fold cross-validation on the training set and evaluate trained models to get the scores of classifiers for both binary and multi-class classification with 88% accuracy score for binary and 77% for multi-class classification [14].

Our research goal is to use data augmentation to improve the performance of Edge/Fog attack detection systems applied to IoT Telehealth systems. This paper presents a novel technique for augmenting data using GAN. Although data augmentation and GAN techniques are used in many application domains, we believe we are the first to use GAN to improve the performance of Edge/Fog attack detection systems

*Ton-IOT telemetry* [1] and *IoTID20* [2] datasets are recently developed and related to our research. We use them because of their different nature and characteristics. The end goal of this thesis is to classify attacks on Edge/Fog Computing-Based attack detection so, we have to train our model based on sensors (edge devices) and communication channel network (Fog computing-based) security to prevent them from IoT attacks.

We observe that realistic attack scenarios are getting difficult to detect. We use cross-evaluation to make the training set strong [3]. We use this approach using these datasets and propose a new framework that is used to conduct an experiment using different classification models.

# Chapter 3

# Background

In this section, the theory and background of the Telehealth systems, IoT devices, IoT ecosystem characteristics, vulnerabilities, security attacks, and aspects are discussed in detail. In addition, it includes machine learning algorithms and evaluation methods that are being used in the report further. Finally, it described the datasets used and information about them.

## 3.1 Telehealth model: Edge & Fog Computing-based system

Edge and Fog computing are related to IoT network systems and is used in IoT telehealth systems as well. We focused primarily to secure the hub of Fog computing, which is Internet cloud using network channel traffic, and Edge computing that would be out IoT edge devices in telehealth system. We saw that these two components in an IoT telehealth system need to be secure to prevent IoT network attacks. Therefore, we used two different nature of datasets that would classify attacks on the telehealth remote model.

In many countries, researchers are trying to develop and enhance the health sector by transforming it into a dynamic, automated, and online medical assistance so health services can help remote communities by diagnosing, accessing, and suggesting individuals the solutions to medical problems that they needed without a hassle. As there are many common growing diseases around us which needed to be rectified as soon as possible. So, in these cases, we always need an online and dynamic automated medical assistance system using telecommunication and the health sector

that can rectify medical problems and suggest possible remedies. Many examples can be seen that patients at home, remote, or in transit could have faster results and save time in an efficient and cost-effective way. The recent example can be seen as COVID-19 reduced physical interaction and prefers online ways to deal with most of the problems. That's why most countries are trying, developing, and researching to achieve that in near future. Many researches are already going on and it's a popular topic these days. The telehealth model is responsible to possess and process the confidential information of patients and it makes this more vulnerable to attackers. The expose of any tiny pipeline can cost confidential data breaches, leading to data freeze or theft. Many authors are discussing those cyber-attacks in their recent research. They further argue that telehealth is one of the most recent technological models that are at risk from cyber-attacks. These essential services can be compromised at any time as they are using IoT devices for communication with the server or the server (databases) that have confidential data [4].

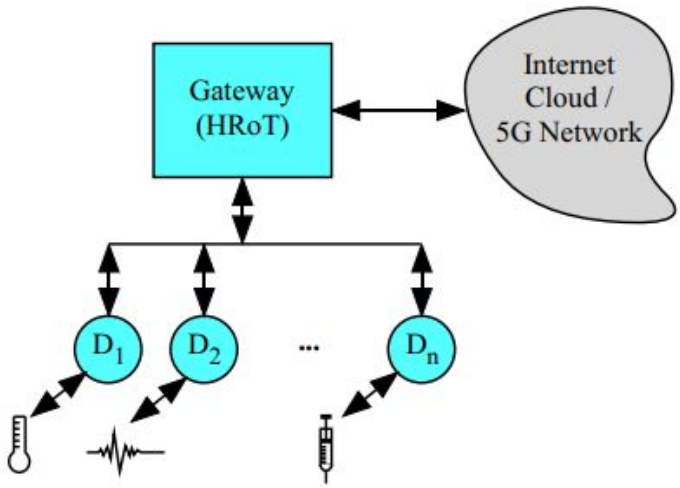


Figure 3.1: Basic IoT telehealth system structure represents client-side [4].

Figure 3.1 shows the representation of the client-side architecture model based on the IoT health care system. It refers to the patient's home as a remote healthcare delivery site. The IoT edge devices ($D_1,\ D_2,\ \cdots, D_n$), using edge computing, are used to gather information to transmit to the gateway $G$. The Gateway focused on securely forwarding the data to the server through the internet cloud. The gateway can be considered as a hardware root-of-trust (HRoT).

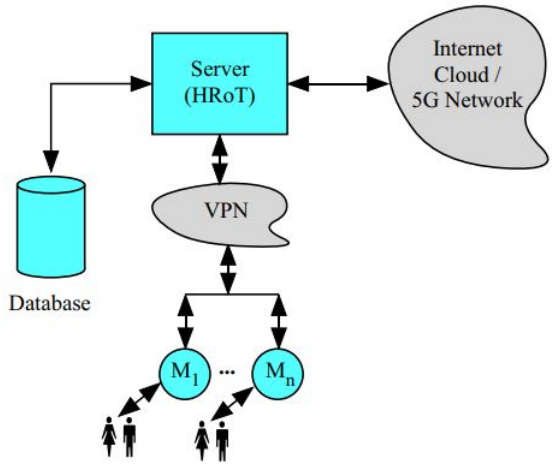


Figure 3.2: Basic IoT telehealth system structure represents server-side [4].

Figure 3.2 is a server-side of the telehealth system. The server-side consists of the hospital and doctors/nurses/administrators that connect to the server through mobile devices. The internet cloud used fog computing and is responsible to process information from the client-side and Mobile devices to the server's database. ($M_1$, $M_2$, $\cdots$, $M_n$). The server is also considered a hardware root-of-trust (HRoT) [4].

## 3.2 Edge Computing (IoT edge devices)

The IoT edge devices are physical devices that could be sensors or actuators. These devices used edge computing to process information to the gateway (HRoT). Examples of sensors could be thermometers, blood glucose meters, or ECG monitors. Examples of actuators could be saline/medicine injection devices or insulin pumps. These devices are typically located in unsecured locations and have limited computer and energy capabilities and might or might not have even a simple operating system. That is the reason behind attackers are targeting these devices.

## 3.3 Fog Computing (Internet Cloud)

The concept of transmitting information from edge computing to another element connected in a network. This is performed to make distant communication from IoT sensors or actuators. This cloud computing component in telehealth systems is also vulnerable to attackers. They can send malicious requests from IoT edge devices and

could damage the database. This cloud computing is performed using Internet cloud connected via LAN or into the LAN hardware itself.

## 3.4 IoT attacks

IoT attacks aim to get access, amend, take control, or corrupt confidential information. The vulnerabilities and limitations of IoT edge devices make them an easy target to different types of attacks.

The most complex and advanced way to attack an IoT network is a botnet attack. A typical botnet is an army of internet-connected devices which works together to launch a targeted attack. These botnet attacks are commonly reported as denial of service (DoS) attacks, thereby consuming system's resources and disrupting operations and services via exhausting all service pipelines [15]. Botnets attacks are hard to detect and are increasing day by day. Attackers are developing more complex variants like ransomware, Mirai, Scanning, or brute force.

### 3.4.1 IoT Ecosystem Characteristics

These devices are designed for dynamic reasons and are often used in ehealth, smartfarms, phones, homes, transportation, and facilities. These devices have different forms of characteristics based on their enhancements. The characteristics have heterogeneity, pervasiveness, mobility, and resource constraints. Figure 3.3 shows the characteristics of an IoT Ecosystem. Those characteristics are extensively described in [16], [17], [18], [19], [20], [21].

Figure 3.3, shows different IoT ecosystem characteristics that are discussed in detail below.

**Pervasive** : The pervasive behavior of IoT devices application provides the functionality which makes them common in any field. These would be everywhere around us in the future. As we are focusing more on automation and robotics to ease human life, IoT would play a vital role that we will rely on more. These devices with dynamic features have many examples like bulbs connected to a motion sensor, parking lights, garden showers, etc. We do often use many other forms of IoT and most people are now dependent on them. This usage must be secured as they are highly related to

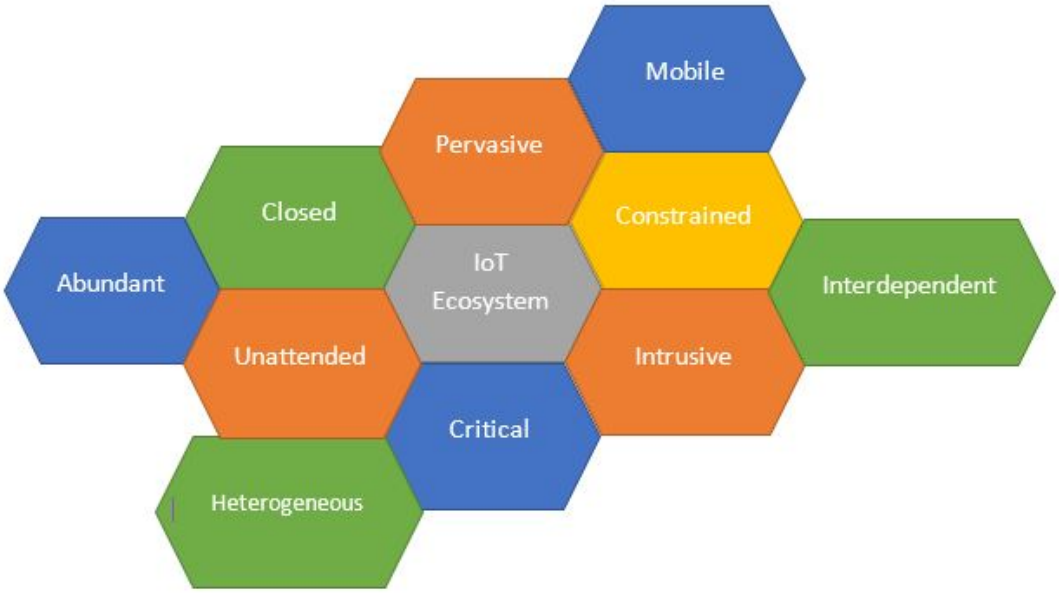


Figure 3.3: IoT ecosystem characteristics.

human and their daily usage. Despite this challenge, people are using it and will use them more because of their way of work.

**Constrained** : The IoT devices are based on resources and their characteristics. They have limited computing, memory, and analyses capabilities. Memory Management Unit (MMU) and address space layout randomization (ASLR) are some components that are not used in some IoT devices. They cannot perform advanced and complex intensive operations on their own. Only encryption and authentication modules are not enough to prevent these devices from malicious attacks and their variants.

**Intrusive** : This characteristic of IoT devices is responsible for gathering personal information like medical devices, wearables heart rate checking devices, blood pressure, location, physical activities, and more. This personal information is really important and needs to be secured as it can be leaked if the channel of communication is not completely secured.

**Critical** : Some areas are sensitive and critical where the availability and performance do matter like medical applications, textile, and agriculture fields. These

applications and their controlling of accessing information are at stake and critical as the usage of IoT devices in them [22].

**Unattended** : IoT devices can work independently for a long time without physical involvement. Research article shows the concern of unattended status for these IoT devices [16]. Hence, without rectifying the unattended behavior of IoT devices, it is really difficult to monitor the statistics and analyze to detect any intrusion.

**Closed** : The closing characteristic shows that IoT is limited to add-on any new software after production. It is *closed* as it's not open to install any security software like antivirus or firewall on the device after manufacture [20].

**Mobile** : The mobility of IoT devices has emerged with IoT innovations. This characteristic concerns the devices that are connected from one network and then move to another network. Once connected to a new network, devices are connected to communicate with other devices. This ability to move from one to another network makes them vulnerable to malware attacks. In addition, devices might get connected to a malicious network and this whole process will propagate malicious activities throughout the network [16].

**Interdependent** : This property in IoT devices shows the accessing and controlling state of other devices. This communication with each other without resorting to physical intervention to their environment. For example, the temperature of a room will decide to start an AC or a heater in a smart home. This whole mechanism can be bypassed by an attacker. Attackers compromise this poor defense configuration that has interdependent relationships with the targeted device. Hence, the interdependent relationships make devices and networks vulnerable.

**Heterogeneous/Diverse** : IoT devices have heterogeneous behavior because of their communication protocols. These protocols making sure the diverseness of IoT devices in different areas like smart grids, e-health, smart industries, and motion sensors. The IoT ecosystem is diverse in many ways and its now deployed in all the major areas and different application fields using communication protocols and hardware platforms. Bluetooth, WiFi, or Z-Wave can be some examples of these communication protocols. This diversity based on communication protocols or hardware-

based-medium (x86, ARM, MIPS, etc.) makes it difficult to implement best practice rules for validations by the system administrator or programmers. It is also not possible to develop a generic security solution for security requirements respective to the hardware platforms, conditions, protocol, or application used.

**Abundant** : Data generated by IoT network devices are abundant. They are rapidly proliferating and day by day growing. In research, it shows more than 41.6 billion connected IoT devices, generating around 79.4 data in zettabytes (ZB) by 2025 [23]. As IoT devices are not protected, leads the information needs to be secured. If information is not secured then it leads attackers to leverage abundant data generation capabilities.

### 3.4.2 Vulnerabilities in the IoT

After discussing the characteristics, a general lack of security threat can be found from both entities including manufacturers and the consumers that reflects the vulnerability of IoT network devices. Manufacturers care about the temporary period of their product and customers are unaware of that and the security issues that could happen due to it. In IoT devices, manufacturers could ignore the basic security prevention practices like updating software and their passwords. Below are some security vulnerabilities that can be encountered in IoT systems.

**Weak credentials** : An easy, simple, and hardcoded password shows the most common vulnerability in the IoT structure. The default password after supply can be easy to break into the system. Customers don't change credentials often and when they do, the new password might often be guessable due to easy combination. Weak credentials increase the vulnerability of IoT devices and potentially lead to common brute-force attacks. Hence, most manufacturers do not implicate strong passwords with complex combinations [24] [25].

**Backdoors** [26]: These are the loopholes left open by vendors or manufacturers for management or testing reasons. This backdoor passage provides open ports such as Telnet, SSH, or FTP [25]. These can easily be found out through scanning which benefits attackers to get admin privileges for the full control of a device [27].

**Software vulnerabilities** : Software vulnerabilities can be seen in IoT devices like buffer overflow or authentication bypass [26]. This leads to providing full control of the IoT device to attackers. As these devices are diverse and heterogeneous so it's very difficult to implement the best programming rules to prevent these attacks [27].

**Inefficient update policy** : The main thing that can be the main reason is IoT devices not having any set of rules defined for the update policies. These policies could be really important for the timely security solution [27] [25].

**Insecure interfaces** : The interfaces like web or mobile applications that control or manage these IoT devices should be highly secured. Most of the time, these interfaces can easily be hacked through SQL injection, a Cross-site script (XSS), or any data leakage script. These interfaces can be vulnerable to attacks and responsible for information leakage [24].

**Lack of privacy/encryption** : IoT devices do not have any set of rules defined for encryption or do not have any mechanism to implement that. This leads the attackers to hack data transmitted via networks. Eavesdropping or any other wireless attack can happen over the network [27].

**Insufficient authentication and authorization mechanism** : Mechanism of authentication and authorization lacks in IoT devices. Manufacturers and vendors often do not implement these strategies that lead intruders to perform Man-in-the-Middle (MitM) attacks. This problem leads to unauthorized access towards data theft [27].

**Poor physical security** : These devices can be often debug through a USB physical port that can be used to extract sensitive information. These devices rarely implement physical security to prevent physical attacks. This exposed intruders to read and hack systems and forge malicious commands in devices [27] [24].

However, IoT devices and their usage increase the vulnerabilities day by day. The use of weak credentials increases the chance of backdoors, software with updated policies would make devices insecure, no encryption and authorization mechanism and poor physical security are the results of vulnerabilities in IoT devices due to lack of security awareness from both the manufacturers and the customers.

### 3.4.3 Security Aspects

Security aspects of IoT devices have some properties that would help us to understand the security factors of IoT devices. The key goal of IoT devices should have those aspects completely achieved to get the ideal secured network. IoT needs to focus to protect data, as authentic user information can also be found in the data obtained from physical devices [28].

**Confidentiality** : Data confidentiality plays a major role in IoT network security. The management of information while communication and the data generated using IoT devices needs more work in existing techniques to improve security for its confidentiality. Authentic identities are often listed as an important principle to guarantee data confidentiality [29]. It can be achieved with the help of encryption algorithms like converting information into ciphertext and the owner has the key. There can be another way like a two-step authentication procedure, only allowing access after identity verification via verified security components.

**Integrity** : Parameter to evaluate any modification and forced errors in the data from any attack leads towards data integrity. It shows the data should same across the network while communicating through IoT devices or sensors. To make sure the data's integrity, security from these data-changing attacks should be implemented like password-based implementations. Another possible solution could be a hardware mechanism with reliable systems [30].

**Privacy** : Every data protection requires some policies governed by the standards to secure the network and handle those networks carefully. These policies and standards ensure the privacy of data and its management.

**Access control** : This aspect is about access permissions and their management. IoT devices have a task to do and this leads towards processing the information as received. This role-based nature has the advantage of managing rights to access only certain services allocated to them. Microsoft Active Directory is an example to access control schemes based on networks via computers. Any malicious attempts to gain access and corporate keys of the network would access to data is limited to specific areas of the network that are authorized by the keys [31].

Due to the exposed vulnerable nature, IoT devices are the primary target. IoT botnets attacks are evolving daily and are getting more advanced and complex to deal with.

### 3.4.4 IoT Security attacks

IoT attacks aim to get access, amend, take control, or corrupt confidential information. The vulnerabilities and limitations of IoT edge devices make them an easy target to different types of attacks.

The most complex and advanced way to attack an IoT network is a botnet attack. A typical botnet is an army of internet-connected devices which works together to launch a targeted attack. These botnet attacks are commonly reported as distributed denial of service (DDoS) or denial of service (DoS) attacks, thereby consuming the system's resources and disrupting operations and services via exhausting all service pipelines [15]. Botnets attacks are hard to detect and are increasing day by day. Attackers are developing more complex variants like ransomware, Mirai, Scanning, or brute force.

IoT devices can be vulnerable to different types of attacks and exposed to those attacks in different ways which leads to their weakness. Some of the most common attacks are discussed below.

**Denial-of-Service (DoS)** : It happened when authentic requests are not able to access the server resources. It may happen when malicious scripts corrupt the channel of communication. This attack used the flooding technique to make the server unavailable with a large number of requests [14].

**Distributed Denial-of-Service (DDoS)** : A denial of service or DoS attack occurs when authentic users are not able to access the confidential data from the server or network. DoS attacker attacks to make the resources unavailable for the authentic request as well. This distributed attack happens when a large number of malicious or bot systems target the same resource or system with attack packets in a network to make it unavailable [14].

**Scanning** : It is referred to as the attacker who scans smart devices for the information of a network. It scans header ports or flow headers like IP addresses. It is a legit attack to get confidential information. This information could use to perform other types of attacks later on [14].

**Ransomware** : It is an advanced type of attack that resists the legit user request to access services by encrypting them and then asking a price to provide the decryption key. So, the legit user gets access to services again. Hence, ransomware denies authorized access to IoT devices. These locked down or denial of access to IoT applications leads to catastrophic circumstances such as financial losses to organizations. The ransomware attack was performed by the Kali systems [14] [1].

**Cross-Site Scripting (XSS)** : The XSS would happen when an attacker remotely injects arbitrary malicious web scripts such as JavaScript codes. Attackers would run different complex malicious commands on the server in the IoT applications [14].

**Password Cracking Attack** : This attack would also take place using malicious scripts that were developed to simultaneously launch password attacks scenarios against weak security IoT/IIoT devices in the testbed. This leads to the path for an attacker to bypass the authentication protocols and compromise IIoT devices and sensitive data through it [14].

**Mirai** : Mirai is a specialized attack that would happen to certain IoT devices. It took advantage of an unsecured network of IoT devices in an efficient way. They controlled the network by bots or "zombies", often used to launch DDoS attacks. Login password attempts to log in default passwords are an example to launch the bot army.

**Man in the Middle (MITM)** : An eavesdropping attack that interrupts an ongoing conversation or data transfer. This type of attack would happen in such a way where the attackers pretend to be the receivers or senders for the authentic participants to get the information or data from either end. Malicious links would send to the users in a way between a conversation that might not be detected until it is too late. Common abbreviations for a man-in-the-middle attack include MITM, MitM, MiM, and MIM [14].

## 3.5 Machine Learning Methods

We used different machine learning techniques and algorithms along with ensemble methods to perform classification on IoT devices attacks. We used Logistic Regres-

sion (LR), Linear Discriminant Analysis (LDA), k-Nearest Neighbour (kNN), Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), and Support Vector Machine (SVM). Also, we used some ensemble methods like AdaBoost (Ada), XGBoost (XGB), and Gradient Boosting (GB) classifiers. We used many classifiers because some classifiers are better for some classifiers and they can handle and increase the accuracies pretty high by just using some deep learning and good feature engineering techniques.

## 3.6 Evaluation criteria

The primary step after performing classification is to evaluate based on a defined approach. The right metric of evaluation would give us a more suitable classifier among others. For real-time scenarios, we need to choose suitable evaluation criteria for our research. Experimental results reported in Section 5 are based on the following parameters.

A confusion matrix is an $N \times N$ matrix that is used to calculate the performance metric of machine learning models such as recall (R), $F_1$-score ($F_1$), accuracy (A), and precision (P). We used them because they provide comparisons of samples out of True Positives, False Positives, True Negatives, and False Negatives [3]. From it, one can infer the following.

- True Positive (TP): Number of true samples classified as true
- False Positive (FP): Number of false samples classified as true
- True Negative (TN): Number of false samples classified as false
- False Negative (FN): Number of true samples classified as false

The above measures define other performance parameters as follows [32]:

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{3.1}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\mathrm{A} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

The $F_1$-score measures the mean (harmonic) using precision and recall as follows

[32]:

$$F_1 = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{3.4}$$

## 3.7 Datasets

For classification, we selected two IoT network-based publicly available datasets that are ToN-IoT Telemetry dataset [1] and IoTID20 dataset [2]. Each dataset contained malicious attacks over an IoT-based network.

### 3.7.1 ToN-IoT Telemetry dataset

This dataset is developed over a realistic network designed at the IoT Labs and Cyber Range. The dataset is called as ToN-IoT dataset and it includes different data sources to collect IoT and IIoT sensors. It developed over a new testbed network for the industry network and includes IoT and IIoT networks. This dataset consisted of using the testbed to manage the interconnection communication between described three layers of IoT, Cloud, and Edge/Fog systems [1].

### 3.7.2 IoTID20 Dataset

The IoTID20 [2] comprised on raw packet created by the IoTID dataset [33]. Different smart devices including phones and laptops connected to Wi-Fi or a network. Nmap tools were used to simulate samples and have different types of attacks [2].

## 3.8 GAN (Generative Adversarial Networks)

Generative Adversarial Network (GAN) is a generative framework consisting of two entities that are Discriminator and Generator. After analyzing the distribution of the training data, the generator constructs fake new data based on the real dataset. The discriminator will decide if the input data should be added to the training set or is fake data [34]. This distribution has been explained in Figure 3.4.

Figure 3.4 shows the mechanism GAN works and how we implement it accordingly [35]. The models like discriminator and generator are trained iteratively. Initially, the generator generates samples using noise, then the discriminator determines the

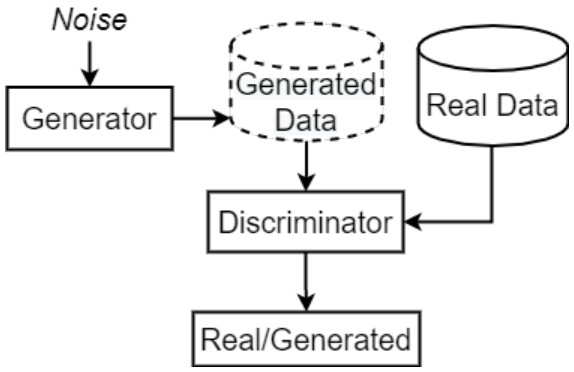


Figure 3.4: Generative Adversarial Networks

samples. Its classification loss is low and the generator loss is consequently high. The goal is to maximize the discriminator loss and minimize the generator loss.

# Chapter 4

# Methodology

In this thesis, we propose a framework that uses different entities to perform classification and detect IoT attacks. We are using a general idea and assuming that it would completely be able to fit for the telehealth model IoT attack classification. This will ensure the security between the devices and the server. This framework will use as a filter to restrict the authorized access and secure the gateway to restrict access to the server.

## 4.1 Framework

Detection of anomalies and malicious activities during the communication between IoT devices and resources is the main function of the modules in Figure 4.1. On the left of the figure is the module that combines the datasets using GAN (Generative adversarial network) approach shown in Section 3.8, to render a balanced dataset. Several classifiers are used, as shown on the right of the figure. The classification results are obtained after a majority voting procedure, shown in Figure 4.1.

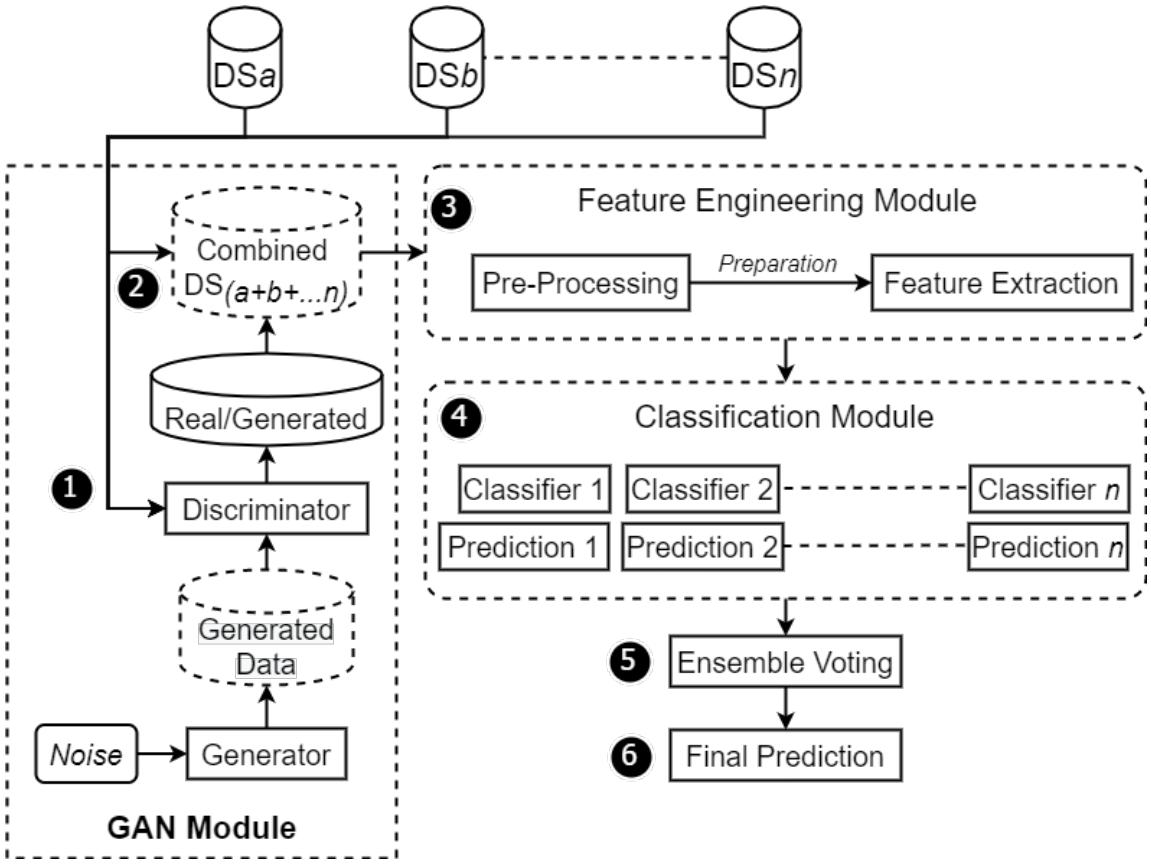


Figure 4.1: Proposed Attack Detection System Framework.

Figure 4.1 shows the framework used in this thesis to achieve the solution to our research problem. Datasets ($D_a$, $D_b$, $\cdots$, $D_n$) used as an input in *Discriminator* to identify real or simulated samples (generated by Generator using Noise). Then, we combined the newly generated dataset and provide it into 'Feature Engineering Module' which is responsible to perform feature engineering and pre-processing on provided dataset. Furthermore, 'Classification Module' took the updated dataset and perform classification using different machine learning techniques to get the final result through voting.

## 4.2 Preparation and pre-processing

Data preparation is the most important step to be done before performing classification. This step holds different approaches that could be performed to make the data is clean and readable for classification. Some important operations involve handling

overfitting and removing extra features (noise) to enhance the performance. Removal of extra spaces if exist using stripping from the right or left value and handling missing values by eliminating the record. Furthermore, normalizing values by taking any measure into account like, mean or standard scalar. We perform these operations on the datasets as explained below in detail.

- Decompressing & Formatting: When we downloaded the publicly available datasets, they were in zip format. So, we extract them into comma-separated values (CSV) Excel format to get an easy python readable form.
- Feature Conversion: Some categorical features that have values in different formats like objects were converted into numeric or integer (0,1) values or in float (decimal) format. The categorical values like labels have been converted to consecutive numeric values by applying a label-encoding method.
- Removal of Extra Features: We removed some features to make classification more accurate and avoid overfitting by unnecessary data.
- Missing Values (NaN): IoTID20 Dataset has huge samples of scanning and DoS attacks. We remove missing or not available values from this dataset. In the ToN-IoT dataset, we used the $Mean$ value of features to replace NaN entries.
- Remove Extra Spaces: We used *.str.strip()* in both datasets to remove all the spaces and extra trailing as we downloaded it from the internet and many values can have some extra padding due to internet scripts. We did that step before label encoding as this would not make any unnecessary encoding using spaces.
- Labelling: We use label encoding in datasets for labels. For binary encoding, we used 0 for 'Normal' and 1 for 'Attack'. For multi-class, we used $0, 1, 2, \cdots$ for classification.
- Cleaning: Duplicate samples are deleted using python scripts. We made some changes to the IoTID20 dataset to make it cleaner. The subcategory was removed because we are only performing binary and category classification. Furthermore, MIRAI samples are excluded because the number of samples was making it complicated. The definition of features used in these two datasets is mentioned below in Table 4.1 and Table 4.2 for ToN-IoT dataset and Table 4.3

and Table 4.4 for IoTID20.

Table 4.1: ToN-IoT Binary Data

| **Binary Label** | **Instances** |
|---|---|
| Anomaly | 245,000 |
| Normal | 156,119 |

We found Anomaly/Attack more then 200,000 samples and around 150,000 in ToN-IoT dataset. It seems balanced after we performed above mentioned pre-processing and feature engineering techniques to make it more clean.

Table 4.2: ToN-IoT Anomaly Data

| **Category Label** | **Instances** |
|---|---|
| Password | 35,000 |
| Injection | 35,000 |
| Backdoor | 35,000 |
| DoS | 25,000 |
| Ransomware | 16,030 |
| XSS | 6,116 |
| Scanning | 3,973 |

We got different attacks in ToN-IoT dataset. We distributed them into number of samples based on each attack. In this thesis, we focused on Normal, DoS and Scanning samples to find out the same attack classification when we merge both datasets.

Table 4.3: IoTID20 Binary Data

| **Binary Label** | **Instances** |
|---|---|
| Anomaly | 265,656 |
| Normal | 27,526 |

We got imbalanced number of records that we balanced after applying pre-processing and feature extraction techniques. We found that this number of selected features are enough to perform classification.

Table 4.4: IoTID20 Anomaly Data

| **Category Label** | **Instances** |
|---|---|
| DoS | 59,362 |
| Scan | 17,766 |
| MITM | 15,044 |

As explained in binary feature distribution for IoTID20, we applied different machine learning techniques to get a clean dataset. We have different attack variants and we are interested to use only DoS, Scanning, and Normal attack types for classification as they are common with another dataset.

### 4.2.1 Data Normalization

Prevention for outweighing is the problem we faced in the datasets used. Some features have larger values than other features and could lead to inaccurate results. We used the normalization technique and make those features ranges between [0,1]. Normalization does not impact the behavior of the features when using min-max technique:

$$z = \frac{(x - x_{min})}{x_{max} - x_{min}} \tag{4.1}$$

Equation 4.1 shows the normalized value $z$ compared to the actual value $x$. Its getting factorized by using minmax subtraction, that is $x\_max$, $x\_min$, after deducting $x\_min$. It usually returns the value in the range of 0.0-1.0. This would be a handy value for the accuracies to avoid any spike. We used Python's `normalize( )` method from `from sklearn import preprocessing` library.

Figure 4.2 shows the result after applied normalization to avoid any bigger values of features in samples of datasets. We applied Python's `normalize( )` method to get simplified features in normal range values to avoid any complications for training models on specified datasets. Also, we improved our accuracies after applying that which shows in Experiment's Section 5. We applied this on each dataset on different features that had huge differences as compared to other samples.

| | | | |
|---|---|---|---|
| 49389 | 52921 | 0.632899 | 0.678160 |
| 49389 | 52921 | 0.632899 | 0.678160 |
| 49389 | 52921 | 0.632899 | 0.678160 |
| 49389 | 52921 | 0.632899 | 0.678160 |
| 40665 | 44748 | 0.555843 | 0.611653 |
| ... | ... | ... | ... |
| 62602 | 1253 | 0.890388 | 0.017821 |
| 62602 | 1253 | 0.890388 | 0.017821 |
| 198 | 38266 | 0.002924 | 0.565111 |
| 198 | 38266 | 0.002924 | 0.565111 |
| 198 | 38266 | 0.002924 | 0.565111 |

Figure 4.2: Applied normalization on ToN-IoT on first two features.

## 4.3 Features Selection

In ToN-IoT the datasets are categorized into different files according to the device like fridge, garage door, GPS tracker, etc. These samples do have some different features based on the device characteristic. We removed *date* and *time* features because there will be always a new *datetime* with respect to the attack. So, there is no point to train models with such samples which have older or no use timestamps. In this work, we combine data of all devices and use the result for the classification. The selected features with their descriptions [14] can be seen in Table 4.5.

Table 4.5: ToN-IoT Feature Description [1]

| Feature | Description |
|---|---|
| FC1 Read Input Register | Modbus function code that is responsible for reading an input register |
| FC2 Read Discrete Value | Modbus function code that is in charge of reading a discrete value |
| FC3 Read Holding Register | Modbus function code that is responsible for reading a holding register |
| FC4 Read Coil | Modbus function code that is in charge of reading a coil |
| Current Temperature | Current temperature reading of a thermostat sensor |
| Door State | State of a door sensor where the door is open or closed |
| Fridge Temperature | Temperature measurement of a fridge sensor |
| Humidity | Temperature measurement of a fridge sensor |

*continued on next page*

| *continued from previous page* | |
|---|---|
| Feature | Description |
| Label | Identify normal and attack records, where '0' indicates normal and 1 indicates attacks |
| Latitude | Latitude value of GPS tracker sensor |
| Light Status | Status of a light sensor is either on or off |
| Longitude | longitude value of GPS tracker sensor |
| Motion Status | status of a motion sensor is either (on or off) where 1 indicates 'ON' and 0 refers to 'OFF' |
| Pressure | Pressure reading of a weather sensor |
| Sphone Signal | Status of receiving the door signal on a phone where 'signal' is true or false |
| Temp Condition | Temperature conditions of a fridge sensor, which temperature is high or low based on a predefined threshold value |
| Temperature | Temperature measurements of a weather sensor |
| Thermostat Status | Status of a thermostat sensor is either on or off |
| Type | A tag with normal or attack sub-classes |

The dataset IoTID20 has 85 features. We performed correlation technique and removed features from IoTID20 dataset. We used a correlation coefficient value of 0.70 [13] to remove features listed in Table 4.6.

Table 4.6: IoTID20 features removed due to correlation

| Total features | Feature name |
|---|---|
| 12 | Active Max, Bwd IAT Max, Bwd Seg Size Avg, Fwd IAT Max, Fwd Seg Size Avg, Idle Max, PSH Flag Cnt, Pkt Size Avg, Subflow Bwd Byts, Subflow Bwd Pkts, Subflow Fwd Byts, Subflow Fwd Pkts |

In addition, we removed extra features such as port numbers, IP addresses, and timestamps because these are related to flow-identifiers and it can be changed with respect to location and time. Attackers are using recent bouncing routers that are helping them to remain untraceable if the system is trained with already used or attacked IP addresses or flow-identifiers. These features will not make the model generalized when deployed as attackers can change IP addresses and will attack at different intru-

sion times. Moreover, 10 more features were deleted as they are single-valued. These features could not impact much when the dataset includes them while training. These are usually either zero or one. So, it would not be going to add any value if we trained our models including these features. We can say that they are independent of attack samples. The features deleted are mentioned in Table 4.7 below.

Table 4.7: IoTID20 Deleted Features

| Total features | Feature name |
|---|---|
| 10 | Flow ID, Src IP, Dst IP, Src Port, Timestamp, Dst Port, Fwd PSH Flags, Fwd URG Flags, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg, Init Fwd Win Byts, Fwd Seg Size Min |

Those finalized features for IoTID20 are listed in Table 4.8 with their description. They are comprised of network-related information. This publicly available dataset is a detailed specification for network-related characteristics of any IoT device.

Table 4.8: IoTID20 Feature Description [2] [3]

| Feature | Description |
|---|---|
| Protocol | Internet Protocol used |
| Flow duration | Duration of the flow in the range of Microsecond |
| total Fwd Packet | Forward direction total packets |
| total Bwd packets | Total packets in the backward direction |
| total Length of Fwd Packet | Total size of packets in the forward direction |
| total Length of Bwd Packet | Total size of packets in backward direction |
| Fwd Packet Length Min | Minimum size of packets in the forward direction |
| Fwd Packet Length Max | Maximum size of packets in the forward direction |
| Fwd Packet Length Mean | Mean size of packets in the forward direction |
| Fwd Packet Length Std | Standard deviation size of packets in the forward direction |
| Bwd Packet Length Min | Minimum size of packets in backward direction |
| Bwd Packet Length Max | Maximum size of packets in backward direction |
| Bwd Packet Length Mean | Mean size of packets in backward direction |

*continued on next page*

*continued from previous page*

| Feature | Description |
|---|---|
| Bwd Packet Length Std | Standard deviation size of packets in backward direction |
| Flow Bytes/s | Number of flow bytes per second |
| Flow Packets/s | Number of flow packets per second |
| Flow IAT Mean | Meantime between two packets sent in the flow |
| Flow IAT Std | Standard deviation time between two packets sent in the flow |
| Flow IAT Max | Maximum time between two packets sent in the flow |
| Flow IAT Min | Minimum time between two packets sent in the flow |
| Fwd IAT Min | Minimum time between two packets sent in the forward direction |
| Fwd IAT Max | Maximum time between two packets sent in the forward direction |
| Fwd IAT Mean | Meantime between two packets sent in the forward direction |
| Fwd IAT Std | Standard deviation time between two packets sent in the forward direction |
| Fwd IAT Total | Total time between two packets sent in the forward direction |
| Bwd IAT Min | Minimum time between two packets sent in the backward direction |
| Bwd IAT Max | Maximum time between two packets sent in the backward direction |
| Bwd IAT Mean | Meantime between two packets sent in the backward direction |
| Bwd IAT Std | Standard deviation time between two packets sent in the backward direction |
| Bwd IAT Total | Total time between two packets sent in the backward direction |
| Bwd PSH Flags | Number of times the PSH flag was set in packets traveling in the backward direction (0 for UDP) |
| Bwd URG Flags | Number of times the URG flag was set in packets traveling in the backward direction (0 for UDP) |
| Fwd Header Length | Total bytes used for headers in the forward direction |
| Bwd Header Length | Total bytes used for headers in the backward direction |
| FWD Packets/s | Number of the forward packets per second |
| Bwd Packets/s | Number of backward packets per second |
| Packet Length Min | Minimum length of a packet |
| Packet Length Max | Maximum length of a packet |
| Packet Length Mean | Mean length of a packet |
| Packet Length Std | Standard deviation length of a packet |
| Packet Length Variance | Variance length of a packet |
| FIN Flag Count | Number of packets with FIN |

*continued on next page*

*continued from previous page*

| Feature | Description |
|---|---|
| SYN Flag Count | Number of packets with SYN |
| RST Flag Count | Number of packets with RST |
| PSH Flag Count | Number of packets with PUSH |
| ACK Flag Count | Number of packets with ACK |
| URG Flag Count | Number of packets with URG |
| CWR Flag Count | Number of packets with CWR |
| ECE Flag Count | Number of packets with ECE |
| down/Up Ratio | Download and upload ratio |
| Average Packet Size | Average size of packets |
| Fwd Segment Size Avg | Average size observed in the forward direction |
| Bwd Segment Size Avg | Average number of bytes bulk rate in the backward direction |
| Fwd Bytes/Bulk Avg | Average number of bytes bulk rate in the forward direction |
| Fwd Packet/Bulk Avg | Average number of packets bulk rate in the forward direction |
| Fwd Bulk Rate Avg | Average number of bulk rate in the forward direction |
| Bwd Bytes/Bulk Avg | Average number of bytes bulk rate in the backward direction |
| Bwd Packet/Bulk Avg | Average number of packets bulk rate in the backward direction |
| Bwd Bulk Rate Avg | Average number of bulk rate in the backward direction |
| Subflow Fwd Packets | The average number of packets in a sub-flow in the forward direction |
| Subflow Fwd Bytes | The average number of bytes in a sub-flow in the forward direction |
| Subflow Bwd Packets | The average number of packets in a sub-flow in the backward direction |
| Subflow Bwd Bytes | The average number of bytes in a sub-flow in the backward direction |
| Fwd Init Win bytes | The total number of bytes sent in the initial window in the forward direction |
| Bwd Init Win bytes | The total number of bytes sent in the initial window in the backward direction |
| Fwd Act Data Pkts | Count of packets with at least 1 byte of TCP data payload in the forward direction |
| Fwd Seg Size Min | Minimum segment size observed in the forward direction |
| Active Min | Minimum time a flow was active before becoming idle |
| Active Mean | Mean-time a flow was active before becoming idle |
| Active Max | Maximum time a flow was active before becoming idle |

*continued on next page*

| *continued from previous page* | |
|---|---|
| Feature | Description |
| Active Std | Standard deviation time a flow was active before becoming idle |
| Idle Min | Minimum time a flow was idle before becoming active |
| Idle Mean | Mean-time a flow was idle before becoming active |
| Idle Max | Maximum time a flow was idle before becoming active |
| Idle Std | Standard deviation time a flow was idle before becoming active |
| Label | Anomaly or Normal |
| Cat | Category of attack or Normal |

It's the challenging and most important stage before getting into classification. As we are using two datasets and will do some experiments after combining and before combining them. So, we need to make sure not to engage more complexity in it as we are dealing with huge datasets. We also don't want to have any overfitting while working on different models and end up without grooming the results. For that, we used the cross-validation method. At first, we used validated selected train and test data and train the models then we used different parameters and used train/test new pair after cross-validation and used those sets to retrain the model again. That gives us more accuracies and we can identify the overfitting in our datasets by that. We use many iterations and used the groomed train/test sets for the model.

# Chapter 5

# Experimental setup & Results

In this chapter, we demonstrate the experimental results for classification using different machine learning algorithms. We used different combinations using the extracted features for each and combined datasets to get the best accuracies while performing binary and multi-class classification.

The accuracy (A), precision (P), recall (R), and $F_1$-score ($F_1$) average scores were calculated for each model on each dataset. StratifiedKFold with 5 and 10 k values cross-validation and hyperparameter tuning were performed within the range of 1-100 input numbers for different classifiers on each dataset. Like for KNN, we used 1 to 100 odd values for `n_neighbors` and performed experiments within the range and saved accuracies in a dictionary respective to their `n_neighbors` value. We did this to get the best `n_neighbors` to determine which parameter is giving us the best accuracy for the specific classifier.

## 5.1 Experiment 1: Ton-IoT dataset

In this experiment, we apply the classification using training and testing samples from Ton-IoT dataset. We perform binary and multi-class classifications.

### 5.1.1 Benign & Malicious Classification

For benign and malicious attack classification, we considered *Label* as the class feature. The experimental analysis of the results is based on datasets, shown in Section 3.7. Table 5.1, shows the classification results in detail.

Table 5.1: Benign & Malicious Classification with ToN-IoT dataset

| | | LR | LDA | KNN | RF | DT | NB | SVM | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results [14] | A | 0.61 | 0.68 | 0.84 | 0.85 | **0.88** | 0.62 | 0.61 | N/A | N/A | N/A |
| | P | 0.37 | 0.74 | 0.85 | 0.87 | 0.90 | 0.63 | 0.37 | N/A | N/A | N/A |
| | R | 0.61 | 0.68 | 0.84 | 0.85 | 0.88 | 0.62 | 0.61 | N/A | N/A | N/A |
| | $F_1$ | 0.46 | 0.62 | 0.84 | 0.85 | 0.88 | 0.51 | 0.46 | N/A | N/A | N/A |
| Our Results | A | 0.62 | 0.71 | 0.90 | **0.91** | 0.90 | 0.74 | 0.61 | 0.82 | 0.90 | 0.79 |
| | P | 0.62 | 0.82 | 0.88 | 0.89 | 0.90 | 0.98 | 0.73 | 0.82 | 0.90 | 0.87 |
| | R | 0.60 | 0.55 | 0.91 | 0.89 | 0.88 | 0.49 | 0.28 | 0.82 | 0.89 | 0.68 |
| | $F_1$ | 0.61 | 0.66 | 0.90 | 0.89 | 0.89 | 0.69 | 0.40 | 0.82 | 0.89 | 0.76 |

The Outcome of Table 5.1, we observe that the best accuracy performance is obtained by the RandomForest classifier. Many classifiers also performed well. The performance measures were obtained using Python's `cross_validate( )` method for getting those from formula. After using StratifiedKFold on the train-test split set, we further prune some parameters and get the best score in terms of evaluation parameters. We compared the results with the existing results and found out that Random Forest, Decision Tree, and XGBoost performed well.

### 5.1.2 Multi-Class Classification

We used *Category* column as the class feature to perform multi-class classification. These experiments with the results on the Ton-IoT dataset. Table 5.2 shows the classification results in detail.

Table 5.2: ToN-IoT dataset Multi-Class Classification

| | | LR | LDA | KNN | RF | DT | NB | SVM | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results [14] | A | 0.61 | 0.62 | 0.72 | 0.71 | **0.77** | 0.54 | 0.60 | N/A | N/A | N/A |
| | P | 0.38 | 0.46 | 0.71 | 0.69 | 0.77 | 0.59 | 0.37 | N/A | N/A | N/A |
| | R | 0.62 | 0.63 | 0.73 | 0.72 | 0.77 | 0.51 | 0.61 | N/A | N/A | N/A |
| | $F_1$ | 0.47 | 0.51 | 0.70 | 0.67 | 0.75 | 0.52 | 0.46 | N/A | N/A | N/A |
| Our Results | A | 0.62 | 0.61 | 0.73 | 0.75 | **0.78** | 0.54 | 0.57 | 0.65 | 0.73 | 0.61 |
| | P | 0.59 | 0.58 | 0.74 | 0.75 | 0.78 | 0.55 | 0.60 | 0.64 | 0.72 | 0.67 |
| | R | 0.59 | 0.58 | 0.73 | 0.74 | 0.73 | 0.53 | 0.53 | 0.63 | 0.71 | 0.61 |
| | $F_1$ | 0.55 | 0.54 | 0.72 | 0.74 | 0.71 | 0.47 | 0.46 | 0.62 | 0.73 | 0.57 |

As in Table 5.2, we observe that the best accuracy performance is obtained by the Decision Tree classifier. We used different models and found that the Decision Tree classifier using ToN-IoT dataset provides the best accuracy. The performance measures were obtained using Python's `cross_validate( )` method. We used a train-test split of 80-20% ratio and used 5 and 10 StratifiedKFold, we further prune some parameters and get the best score in terms of evaluation parameters. We also compared the results with the existing one and our approach provides us satisfying results. We also used more boosting algorithms to add new comparative classifier results.

## 5.2 Experiment 2: IoTID20 dataset

In this experiment, we performed classification using IoTID20 training and testing samples. We followed the same approach that we discussed earlier. We performed binary and multi-class classifications in each iteration.

### 5.2.1 Benign & Malicious Classification

Benign and malicious attacks classification considered *Label* as the class feature. The results in Table 5.3, show the classification results in detail.

Table 5.3: IoTID20 dataset Benign and Malicious Classification

| | | LDA | KNN | RF | DT | SVM | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results [36] | A | 0.79 | 0.97 | 0.97 | **0.98** | 0.83 | 0.91 | N/A | N/A | N/A | N/A |
| Our Results | A | 0.93 | 0.97 | 0.97 | **0.98** | 0.83 | 0.93 | 0.44 | 0.97 | **0.98** | 0.97 |

As in Table 5.3, we observe that the best accuracy performance is obtained by Decision Tree and XGBoost classifiers. We used different models and found that Decision Tree and XGBoost classifiers using the IoTID20 dataset provide the best accuracy. We also compared our results with recent work accuracies and found our approach satisfied. The performance measures were obtained using Python's `cross_validate( )` method. After using StratifiedKFold on the train-test set of 80-20% ratio, we further did some pruning to parameters and get the best score in terms of evaluation parameters.

### 5.2.2 Multi-Class Classification

For multi-class classification, *Category* would be the class feature. These experiments were performed on IoTID20 dataset using training and testing samples from the same dataset.

Table 5.4: IoTID20 dataset Multi-Class Classification

| | | LDA | KNN | RF | DT | SVM | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results | A | 0.84 | 0.89 | 0.91 | **0.92** | 0.79 | 0.82 | 0.67 | 0.82 | **0.92** | 0.91 |
| | P | 0.82 | 0.88 | 0.90 | 0.91 | 0.78 | 0.79 | 0.72 | 0.79 | 0.91 | 0.90 |
| | R | 0.77 | 0.86 | 0.87 | 0.88 | 0.73 | 0.75 | 0.71 | 0.79 | 0.88 | 0.87 |
| | $F_1$ | 0.77 | 0.86 | 0.88 | 0.89 | 0.73 | 0.75 | 0.66 | 0.79 | 0.90 | 0.88 |

As in Table 5.4, the best performance can be seen with the accuracy result of 91% obtained by using Decision Tree and XGBoost classifiers. Experimented with many models and found that the Decision Tree and XGBoost classifiers using the IoTID20 dataset performed well. The performance measures were obtained using Python's

`cross_validate( )` method. After using StratifiedKFold on the train-test split of 80-20% ratio, we further prune some parameters and get the best score in terms of evaluation parameters.

## 5.3 Experiment 3: Combined ToN-IoT & IoTID20 datasets

In this experiment, we used our model and combined both datasets to train different machine learning models and then test the newly combined dataset with each of them separately. To combine, we need to analyze each dataset based on attack and non-attack samples.

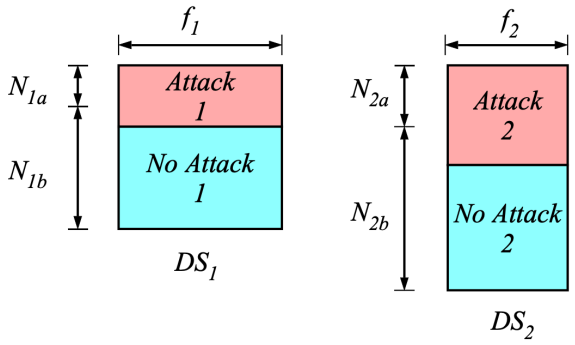


Figure 5.1: Representation of datasets based on Attacks and Normal samples [5]

As shown in Fig. 5.1, IoTID20 dataset $DS_1$ is divided into Attack samples $N_{1a}$ and no-attack samples $N_{1b}$ with features $F_1$. Similarly, ToN-IoT dataset $DS_2$ is divided into Attack samples $N_{2a}$ and no-attack samples $N_{2b}$ with features $F_2$.

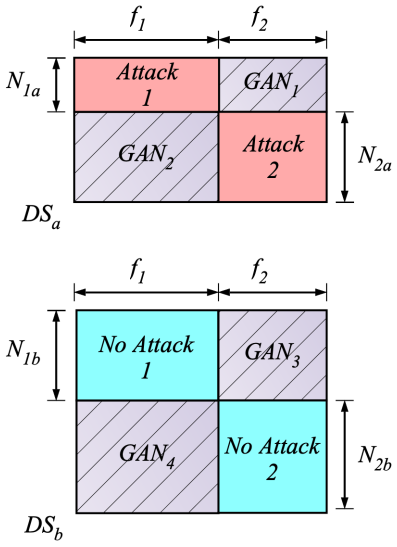


Figure 5.2: Combined dataset using GAN's simulated data by real datasets [5]

As shown in Fig. 5.2, for the binary classification, GANs are used to augment and merge attack samples datasets $DS_1$ and $DS_2$ to generate the attack samples of dataset $DS_a$. For attack samples $N_{1a}$, we used $\text{GAN}_2$ to generate data whose features are $f_1$. For attack samples $N_{2a}$, we used $\text{GAN}_1$ to generate data whose features are $f_2$.

Similarly, GANs are used to augment and merge no-attack samples of datasets $DS_1$ and $DS_2$ to generate dataset $DS_b$. For no-attack samples $N_{1b}$, we used $\text{GAN}_4$ to generate data whose features are $f_1$. For no-attack samples $N_{2b}$, we used $\text{GAN}_3$ to generate data whose features are $f_2$.

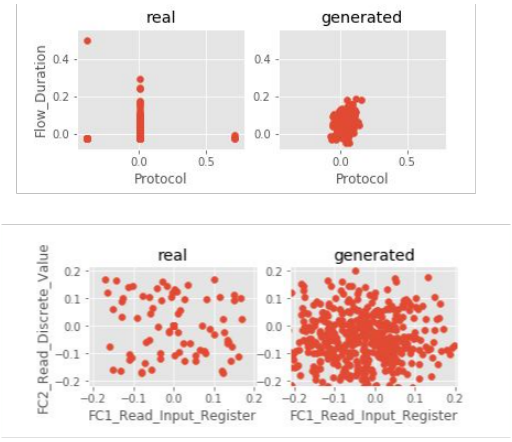


Figure 5.3: GAN generated samples of single feature from each dataset

We generated samples based on GAN logic shown in Fig. 5.3. We used Python's

library GAN_171103.py [37]. This explains one section of iteration based on one feature.

### 5.3.1 Experiment 3a

Here we use the combined dataset as a training set and IoTID20 i.e $DS_a$ dataset for testing the classifiers. We first perform Benign and Malicious attacks and then multi-class category-based classification.

#### 5.3.1.1 Benign & Malicious Classification

Benign and Malicious attacks considered *Label* as the class feature. The results in Table 5.5 show the classification results in detail.

Table 5.5: Benign & Malicious attacks on DS for training and IoTID20 $DS_a$ dataset

| | | LDA | KNN | RF | DT | SVM | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results | A | 0.62 | 0.60 | 0.84 | **0.87** | 0.62 | 0.54 | 0.70 | 0.67 | 0.68 | 0.68 |
| | P | 0.59 | 0.86 | 0.87 | 0.89 | 0.88 | 0.55 | 0.79 | 0.66 | 0.64 | 0.65 |
| | R | 0.59 | 0.58 | 0.91 | 0.95 | 0.87 | 0.53 | 0.85 | 0.73 | 0.80 | 0.80 |
| | $F_1$ | 0.55 | 0.70 | 0.90 | 0.91 | 0.70 | 0.47 | 0.82 | 0.69 | 0.71 | 0.70 |

As in Table 5.5, the best performance can be seen with the accuracy given by the Decision Tree of 87% using above mentioned steps and approach. Some other classifiers also did perform well. After merging the IoTID20 dataset to a different dataset, accuracies get almost the same and some classifiers show a bit low scores as well. To train the model for the real-time attacks we have to process it and train it to classify IoT edge devices attacks as well as fog computing network attacks. After using StratifiedKFold, we further used hyper-parameter tuning to get the best parameters and get the best score in terms of evaluation parameters out of it.

#### 5.3.1.2 Multi-Class classification

*Category* would be used as the class feature for multi-class classification. These experiments were performed on the IoTID20 dataset using training and testing samples

from the same dataset.

Table 5.6: Multi-class Classification with DS for training and IoTID20 $DS_a$ dataset

| | | LDA | KNN | RF | DT | SVM | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Our Results | A | 0.79 | 0.80 | 0.81 | **0.85** | 0.58 | 0.61 | 0.56 | 0.62 | 0.81 | 0.79 |
| | P | 0.77 | 0.79 | 0.85 | 0.88 | 0.58 | 0.56 | 0.57 | 0.62 | 0.81 | 0.79 |
| | R | 0.71 | 0.77 | 0.87 | 0.77 | 0.53 | 0.58 | 0.56 | 0.62 | 0.81 | 0.79 |
| | $F_1$ | 0.71 | 0.78 | 0.81 | 0.78 | 0.53 | 0.58 | 0.47 | 0.62 | 0.81 | 0.79 |

As in Table 5.6, the best performance can be seen with the accuracy given by the Decision Tree of 85% using above mentioned steps and approach. After performing classification, we trained the model for the real-time attacks which is why we are having a bit change in accuracy scores. We trained our model to classify based on IoT edge devices attacks as well as fog computing network attacks. After using StratifiedKFold, we further used hyper-parameter tuning to get the best parameters and get the best score in terms of evaluation parameters out of it.

### 5.3.2 Experiment 3b

In this section, we used the combined dataset as a training set and ToN-IoT i.e $DS_b$ dataset for testing the classifiers. We first perform two-class benign and malicious attacks classification and then multi-class category-based classification.

#### 5.3.2.1 Benign & Malicious Classification

Benign and Malicious attacks classification are considered *Label* as the class feature. The results in Table 5.7, shows the classification results in detail.

Table 5.7: Benign and Malicious Classification with DS for training and ToN-IoT $DS_b$ dataset

| | | LDA | KNN | RF | DT | LR | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results | A | 0.58 | 0.57 | 0.77 | **0.79** | 0.58 | 0.49 | 0.61 | 0.64 | 0.60 | 0.60 |
| | P | 0.57 | 0.48 | 0.63 | 0.77 | 0.58 | 0.46 | 0.75 | 0.63 | 0.46 | 0.43 |
| | R | 0.57 | 0.71 | 0.46 | 0.75 | 0.58 | 0.44 | 0.77 | 0.70 | 0.19 | 0.21 |
| | $F_1$ | 0.55 | 0.57 | 0.52 | 0.76 | 0.58 | 0.47 | 0.73 | 0.68 | 0.27 | 0.29 |

As in Table 5.7, the best performance can be seen with the accuracy given by the Decision Tree of 79% using above mentioned steps and approach. Some other classifiers also did perform well. After merging the ToN-IoT dataset to a different dataset, accuracies get almost the same and some classifiers show a bit change in scores as well. To train the model for the real-time attacks we have to process it and train it to classify IoT edge devices attacks as well as fog computing network attacks. After using StratifiedKFold, we further used hyper-parameter tuning to get the best parameters and get the best score in terms of evaluation parameters out of it.

#### 5.3.2.2 Multi-Class Classification

For multi-class classification, *Category* is taken as the Label feature to perform classification. These experiments were performed on the ToN-IoT dataset using training and testing samples from the same dataset.

Table 5.8: Multi-class Classification with DS for training and ToN-IoT $DS_b$ dataset

| | | LDA | KNN | RF | DT | LR | MLP | NB | Ada | XGB | GB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Results | A | 0.62 | 0.55 | 0.72 | **0.73** | 0.65 | 0.62 | 0.72 | 0.65 | 0.68 | 0.61 |
| | P | 0.59 | 0.52 | 0.72 | 0.73 | 0.65 | 0.59 | 0.72 | 0.64 | 0.68 | 0.58 |
| | R. | 0.59 | 0.52 | 0.72 | 0.73 | 0.65 | 0.60 | 0.72 | 0.64 | 0.68 | 0.57 |
| | $F_1$ | 0.55 | 0.53 | 0.72 | 0.73 | 0.65 | 0.57 | 0.72 | 0.65 | 0.68 | 0.58 |

As in Table 5.8, the best performance can be seen with the accuracy given by the Decision Tree of 73% using above mentioned steps and approach. After performing classification, we trained the model for the real-time attacks which is why we are having a bit different accuracy scores. We trained our model to classify based on IoT edge devices attacks as well as fog computing network attacks. After using StratifiedKFold, we further used hyper-parameter tuning to get the best parameters and get the best score in terms of evaluation parameters out of it.

# Chapter 6

# Conclusion and Future Work

In this work, we addressed the problem of IoT telehealth security attack detection. We achieved good results after performing multiple experiments using different combinations on relevant datasets and classified attacks based on their nature. We have proposed a hybrid model and a framework which trained the machine learning models and is practical to work in real-time applications.

We applied different machine learning algorithms with k-fold cross-validation to obtain the best result and evaluated their performance based on accuracy, precision, and recall via different iterations. To develop a strong detection system, it is a primary and important role to train the models using multiple datasets and get different results. So, we performed different experiments on each dataset and also perform operations after combining them using GAN logic to make a compatible training set. We performed cross-evaluation after testing with each dataset on augmented trained data models and get the accuracies for the system. These results show decent and promising accuracies. A bit of accuracy drop can be seen compared to the original dataset accuracy score but our model is novel and classifying both types like edge and fog computing attacks. This shows the promising solution towards telehealth application using machine learning techniques in classifying threats on IoT networks.

Our novel framework used publicly available datasets which we assumed for telehealth systems. There are several steps for future work. Below are some directions:

- Extend proposed framework by making it as an adaptive model. So, it could improve itself respective to the problem given.
- We focused on telehealth systems for IoT security from attacks. It can be used for different fields as this is a novel approach and after making it adaptive it

can improve itself and train different natures of more datasets to make them compatible and balanced and could solve the classification problem.

- We used publicly available datasets and perform classification on them. A new dataset could be introduced by scrapping or creating using IoT testbeds to secure edge/fog computing in any different area.
- Extend this framework to use for advanced real-time applications working with distributed techniques like distributed databases or topologies to secure the channel in real-time communication.

# Bibliography

[1] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "The TON_IoT Datasets," [Online]. Available: `https://research.unsw.edu.au/projects/toniot-datasets`.

[2] I. Ullah and Q. H. Mahmoud, "IoT Intrusion Dataset 2020," [Online]. Available: `https://sites.google.com/view/iot-network-intrusion-dataset/home`.

[3] A. Farah, *Cross Dataset Evaluation for IoT Network Intrusion Detection*. PhD thesis, The University of Wisconsin-Milwaukee, 2020.

[4] M. Fakroon, F. Gebali, and M. Mamun, "Multifactor authentication scheme using physically unclonable functions," *Internet of Things*, vol. 13, pp. 1–28, 2021.

[5] F. Gebali, *GAN on Different datasets, Private communication*.

[6] S. S. Swarna Sugi and S. R. Ratna, "Investigation of machine learning techniques in intrusion detection system for IoT network," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 1164–1167, 2020.

[7] T. Phu, L. Hoang, N. Toan, N. Tho, and N. Binh, "CFD-Vex: A novel feature extraction method for detecting cross-architecture IoT malware," in *SoICT 2019: Proceedings of the Tenth International Symposium on Information and Communication Technology*, pp. 248–254, Dec 2019.

[8] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[9] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[10] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *ICISSp*, pp. 253–262, 2017.

[11] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pp. 407–414, 2016.

[12] S. Ustebay, Z. Turgut, and M. A. Aydin, "Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 71–76, 2018.

[13] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in IoT networks," in *Advances in Artificial Intelligence* (C. Goutte and X. Zhu, eds.), pp. 508–520, Springer International Publishing, 2020.

[14] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020.

[15] M. Mehra, J. N. Paranjape, and V. J. Ribeiro, "Improving ml detection of IoT botnets using comprehensive data and feature sets," in *2021 International Conference on COMmunication Systems NETworkS (COMSNETS)*, pp. 438–446, 2021.

[16] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606–1616, 2019.

[17] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities," in *2014 IEEE*

*7th International Conference on Service-Oriented Computing and Applications*, pp. 230–234, 2014.

[18] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[19] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[20] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2018.

[21] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 336–341, 2015.

[22] C. Mortimer, "Hackers are now able to take control of cars to cause deliberate accidents, scientists warn," 2017. [Online]. Available: `https://www.independent.co.uk/life-style/gadgets-and-tech/news/computer-hackers-control-car-deliberate-accidents-national-security-issue-\\a8066466.html/`.

[23] L. F. Rembert, "Connected devices will generate 79 zettabytes of data by 2025," 2020. [Online]. Available: `https://iotbusinessnews.com/2020/08/10/08984-connected-devices-will-generate-79-zettabytes-of-data-by-2025`.

[24] P. Ferrara, A. K. Mandal, A. Cortesi, and F. Spoto, "Static analysis for discovering iot vulnerabilities," *International Journal on Software Tools for Technology Transfer*, vol. 23, no. 1, pp. 71–88, 2021.

[25] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the IoT zombie armies," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, pp. 267–272, 2017.

[26] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*, pp. 230–234, IEEE, 2014.

[27] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[28] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in *2015 IEEE symposium on computers and communication (ISCC)*, pp. 180–187, IEEE, 2015.

[29] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[30] D. M. Mendez, I. Papapanagiotou, and B. Yang, "Internet of things: Survey on security and privacy," *arXiv preprint arXiv:1707.01879*, 2017.

[31] A. Jurcut, T. Niculcea, P. Ranaweera, and N.-A. Le-Khac, "Security considerations for internet of things: A survey," *SN Computer Science*, vol. 1, Jun 2020.

[32] M. K. Elhadad, K. F. Li, and F. Gebali, "Detecting misleading information on covid-19," *IEEE Access*, vol. 8, pp. 165201–165215, 2020.

[33] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "IoT network intrusion dataset," [Online]. Available: `https://dx.doi.org/10.21227/q70p-q449`.

[34] A. Sethia, R. Patel, and P. Raut, "Data augmentation using generative models for credit card fraud detection," *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[36] M. Omar and L. George, "Toward a lightweight machine learning based solution against cyber-intrusions for IoT," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 519–524, 2021.

[37] C. Nash, "Create data from random noise with generative adversarial networks," [Online]. Available: `https://www.toptal.com/machine-learning/generative-adversarial-networks`.