

Deep Learning in Cyber Security for Internet of Things

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Furkan Yusuf Yavuz

in partial fulfillment for the
degree of Master of Science

in

Cybersecurity Engineering

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Cybersecurity Engineering.

APPROVED BY:

Prof. Dr. Ensar Gül
(Thesis Advisor)

.....
.....

Dr. Devrim Ünal
(Thesis Co-advisor)

.....
.....

Yrd. Doç. Dr. Mehmet Baysan

.....
.....

Yrd. Doç. Dr. Ömer Korçak

.....
.....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL:

2018-01-12

SEAL/SIGNATURE:



Declaration of Authorship

I, Furkan Yusuf Yavuz, declare that this thesis titled, 'Deep Learning in Cyber Security for Internet of Things' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____



Date: _____

12-01-2018

"Everyone tells their own stories."

Mehmet Yavuz



Deep Learning in Cyber Security for Internet of Things

Furkan Yusuf Yavuz

Abstract

Cyber threats are a showstopper for Internet of Things (IoT) which has recently gained popularity. Network layer attacks on IoT can cause significant disruptions and loss of information. Among such attacks, routing attacks are especially hard to defend against because of the ad-hoc nature of IoT systems and resource constraints of IoT devices. Hence a an efficient approach for detecting and predicting IoT attacks is needed. For the security of IoT, detecting malicious attacks is vital to avoid of unintended consequences such as lack of availability, integrity and confidentiality. For secure IoT needs a system that is able to robust detection against routing attacks. We propose a deep-learning based for continuous security monitoring analysis for IoT. Application of deep learning for cyber-security in IoT requires the availability of substantial IoT attack data, however the lack of IoT attack data is an important issue. In our study, the Cooja IoT simulator has been utilized for generation of high-fidelity attack data, within IoT networks ranging from up to 1000 nodes. We propose a highly scalable, deep-learning based attack detection methodology for detection of IoT routing attacks with high accuracy and precision.

Keywords: deep learning, machine learning, internet of things, cyber-physical systems, cyber security, routing attacks

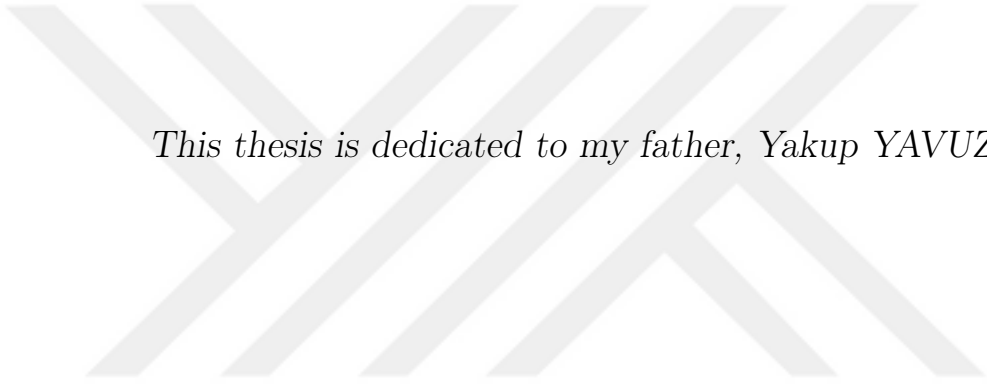
Nesnelerin İnternetinin Siber Güvenliđi için Derin Öğrenme

Furkan Yusuf Yavuz

ÖZ

Son zamanlarda önemi hızla artan nesnelerin internetin için siber saldırıların da önemi hızla artmaktadır. Nesnelerin internetine, ağ katmanında yapılacak saldırılar veri kaybına ve bölünmesine yol açabilmektedir. Siber saldırılar içerisinde yönlendirme saldırıları, nesnelerin internetinin yapısı ve kaynak kısıtları sebebiyle, savunmasını hayli zordur. Bu sebeple nesnelerin internetine yönelik saldırıları tespit edecek bir yöntem ihtiyacı vardır. Nesnelerin internetini izlemek ve analiz etmek, kötücül saldırıları öngörmek, beklenmeyen durumlara ayak uydurmak, önlemler almak, hassas verileri korumak ve kayıpları azaltmak için hayati önem arz etmektedir. Biz, derin öğrenme tabanlı bir güvenlik sistemi sunuyoruz. Derin öğrenme çalışmalarının en önemli kısıtlarından biri olan veri seti eksikliğini gidermek için Cooja simülatörü ile ürettiğimiz ve hazırladığımız veri setini de çalışmamızın ek ürünü olarak sunuyoruz. Veri setimiz 1000'e varan düğümlü kablosuz sensör ağlarını içeriyor. Bunun yanı sıra, ölçeklendirilebilir derin öğrenme tabanlı yönlendirme atak tespit modelleri ile nesnelerin interneti için sağlam bir güvenlik sunuyoruz.

Anahtar Sözcükler: derin öğrenme, nesnelerin interneti, makine öğrenmesi, siber-fiziksel sistemler, siber güvenlik



This thesis is dedicated to my father, Yakup YAVUZ.

Acknowledgments

First of all, i am appreciative to my elder brother Mehmet YAVUZ for his encouragement to pursue my MSc Degree and helping me understand and enrich my ideas.

A special thanks goes to my family for their endless support, especially my dear mother and sister. They have always been there for me whenever I needed it most. Their constant support has always kept me going ahead. I owe them a great deal of gratitude for always being there.

I would like to thank to my dear friend Yusuf KOÇYİĞİT for reading my thesis, commenting on my views and his valuable feedbacks. I also would like to thank all my friends for their best wishes for me.

I would also thank to TÜBİTAK BİLGEM for its opportunities and contribution on my experience on Cyber Security that let me complete my MS study.

Last but not least, a very special thanks goes to my dear grandmother, who motivated and encouraged me to finish this scientific research. This thesis would never be completed without her support.

Contents

Abstract	iv
Öz	v
Acknowledgments	vii
List of Figures	x
List of Tables	xii
Abbreviations	xiii
1 Introduction	1
1.1 Context	1
1.2 Security Problems of Internet of Things	2
1.3 Proposed Methodology in the Thesis	3
1.4 Contributions	7
1.5 Outline	7
2 Background	8
2.1 Internet of Things (IoT)	8
2.1.1 Cyber-Physical Systems (CPS)	8
2.1.2 IoT	10
2.1.3 Challenges of IoT	10
2.1.4 Threats and Risks of IoT	11
2.1.5 Simulation of Routing Attacks	12
2.1.5.1 Routing Attacks to IoT	14
2.2 Deep Learning for Cyber Security	19
2.2.1 Machine Learning	19
2.2.2 Deep Learning	21
3 Related Work	27
3.1 Deep Learning based Cyber Security Methods	34
4 IoT Routing Attack Dataset(IRAD)	36
4.1 Feature Extraction	37
4.2 Feature Normalization	42
4.3 Feature Importance and Selection	43
4.4 Overview of Datasets	46

5	Deep Learning Based Detection of Routing Attacks	48
5.1	Routing Attack Detection	48
5.2	Deep Learning Model	49
6	Evaluation and Conclusion	53
6.1	Performance Evaluation	53
6.2	Analysis	54
6.2.1	Decreased Rank Attack	55
6.2.2	Hello Flood Attack	56
6.2.3	Version Number Attack	58
6.3	Conclusion	60
6.4	Future Works	60
	Appendix	61
	Bibliography	75



List of Figures

1.1	Intrusion Detection Approaches	4
1.2	An overview of IoT attacks	5
1.3	Methodology Flow Diagram	6
2.1	CPS Overview	9
2.2	Components of a CPS	9
2.3	IoT Architecture	11
2.4	System of IoT	12
2.5	Cooja User Interface	13
2.6	A sample of the Simulation	13
2.7	Sample 6LoWPAN Concept	14
2.8	Routing Attacks in IoT	16
2.9	Decreased Rank Attack	17
2.10	Before Hello-flood Attack	18
2.11	After Hello-flood Attack	18
2.12	Brain Cell and Artificial Neuron	23
2.13	Step Function	23
2.14	Sigmoid Function	24
2.15	Tanh Function	24
2.16	ReLU Function	25
3.1	Decision Tree's Logic	28
3.2	AODV Protocol	29
3.3	Sybil Attack	31
3.4	Wormhole Attack	33
4.1	Foren6 User Interface	41
4.2	Transmission Rate before Feature Normalization Process	42
4.3	Transmission Rate after Feature Normalization Process	42
4.4	Feature Importance for Decreased Rank Attack after Feature Selection	45
4.5	Significant Feature Histogram of Version Number Attack Dataset	45
4.6	Insignificant Feature Histogram of Version Number Attack Dataset	45
4.7	Pearson Rate of Total Transmission Time Feature	46
5.1	Deep Neural Network Layers	51
6.1	Model Accuracy of Decreased Rank Dataset	56
6.2	Model Loss of Decreased Rank Dataset	56
6.3	Model Accuracy of Hello-Flood Dataset	57

6.4	Model Loss of Hello-Flood Dataset	58
6.5	Model Accuracy of Version Number Dataset	59
6.6	Model Loss of Version Number Dataset	59



List of Tables

4.1	Details of Datasets	36
4.2	A Sample of Raw Dataset	37
4.3	Encode of Information Feature	39
4.4	Extracted Features	40
4.5	Sample Dataset	41
4.6	Feature Importance of Decreased Rank Attack	44
4.7	Datasets with Numbers	47
4.8	A Comparision of Datasets	47
6.1	Training Performance on Original Dataset	55
6.2	Performance of Decreased Rank Model	55
6.3	Performance of Hello-Flood Model	57
6.4	Performance of Version Number Model	58
6.5	Performance of the Models over Multiple Datasets	60

Abbreviations

CPS	Cyber Physical Systems
IoT	Internet of Things
DL	Deep Learning
ML	Machine Learning
GPU	Graphical Processing Unit
SVM	Support Vector Machines
WSN	Wireless Sensor Networks
DODAGs	Destination Oriented Directed Acyclic Graphs
DIO	Destination oriented directed acyclic graphs Information Object
DAO	Destination oriented directed acyclic graphs Advertisement Object
DIS	Destination oriented directed acyclic graphs Information Solicitation
6LoWPAN	IPv6 over Low-powered Wireless Personal Area Networks
RPL	Routing Protocol for Low-Power and Lossy Networks
AODV	Ad hoc On demand Distance Vector
CSV	Comma Separated Values
PCAP	Packet CAPture

Chapter 1

Introduction

In this chapter, we will explain the problem definition and demonstrate the methodology, objectives of our study, then, briefly mention the contributions. Finally, we will outline the structure.

1.1 Context

We are living in data driven age. Data has been locating (or is going to locate) every point of our life. Most people think that this influence is a consequences of industry 4.0 that makes our life faster than before as all other industrial revolutions [1]. Industry 4.0 enabled the cooperation between computers (or cyber domain) and physical systems. This cooperation is called cyber-physical systems (CPS). Physical system then created the internet of things (IoT) by embedding sensors, controllers, and actuators. Another consequence of the revolution the enormous data that is generated and must be managed, called Big Data.

There are many tools to simulate the IoT environment, consequently Big Data, to study on this innovations easier. Most popular simulators are Cooja [2], GNS-3 [3], Iotify [4] and MATLAB [5]. Unfortunately, to establish successful communication of IoT is difficult point to achieve and maintain. While number of generated data has been increasing, dependently, the term data security has become a crucial term specifically regarding security of sensitive data in alignment with the Three principles of information security (Confidentiality, Integrity and Availability). There are many attacks (car hacking, DDoS

or physical attacks) to IoT because of, particularly, a lack of robust routing protocols. The statistics claim that DDoS attacks increased 91% in 2017 due to the exploitation of IoT devices [6]. IoT, which is in all branches of life, is vulnerable against several types of attacks. However, there is also no effective solution for protecting our life from being affected by these attacks. Nowadays, machine learning(ML) is the most popular study topic for detecting cyber attacks for IoT security. Because, ML based solutions can offer a robust system to unseen attacks. On the contrary, the biggest problem of research on IoT security is the lack of public datasets. Hence there must be comprehensive studies to find solutions in these problems.

1.2 Security Problems of Internet of Things

IoT is under risk due to it's heterogenous structure which in turn enables cooperation of cyber domain and physical domain. Vulnerabilities of IoT are listed by OWASP [7]. Insufficient authentication, insecure network services, lack of transport encryption and integrity verification, privacy concerns, insecure software or firmware, poor physical security are in the list, additionally, insufficient routing protocols can be appended to the list.

We are already at the position of meeting face to face with the consequences of these mentioned vulnerabilities. In October of 2016, the largest DDoS attack was carried out by using IoT botnets. PayPal, The Guardian, Netflix, Reddit and CNN, particularly, became the target. The botnets were made by a malware called Mirai. This malware exploited the security vulnerability of IoT device's login informations. Exploited devices were directed to the targets. Using default username, password, non-unique passwords and lack of software and firmware updates caused the Mirai attack [8].

Full story of Jeep hacking was explained in Black Hat USA 2015 Conference. Researchers hijacked the jeep by exploiting a firmware update vulnerability. They showed that they could make the jeep slow down and speed up [9].

1.3 Proposed Methodology in the Thesis

The term of IoT is a system of interconnected devices, machines and related software services. It has been playing an important role in the modern society since it enabled energy efficient automation for enhancing quality of life. However IoT systems are an obvious target for cyber-attacks because of their ad-hoc and resource-constrained nature. Therefore, continuous monitoring and analysis is needed for securing IoT systems. Because of the vast amount of network and sensing data produced by IoT devices and systems, Big Data and ML methods are highly effective in continuous monitoring and analysis for the security of IoT systems.

In this thesis we propose a highly-scalable deep-learning based attack detection method for realistic IoT scenarios. We have processed data with size close to 64×10^6 . We obtained a high degree of training accuracy (up to 99.5%) and F1-scores (up to 99%). In this study, we have focused on specific IoT routing attacks, namely, decreased rank, version number modification and hello-flood.

Systems security requirements depend to robustness against routing attacks. We propose a deep-learning based for detection for routing attacks to IoT.

There are three main intrusion detection approaches in the literature; misuse detection, anomaly detection, and specification-based detection. Additionally, hybrid-based system is also located under the intrusion detection topic and it's, briefly, a mixture of misuse detection and anomaly detection. Intrusion detection scheme is also depicted in Figure 1.1.

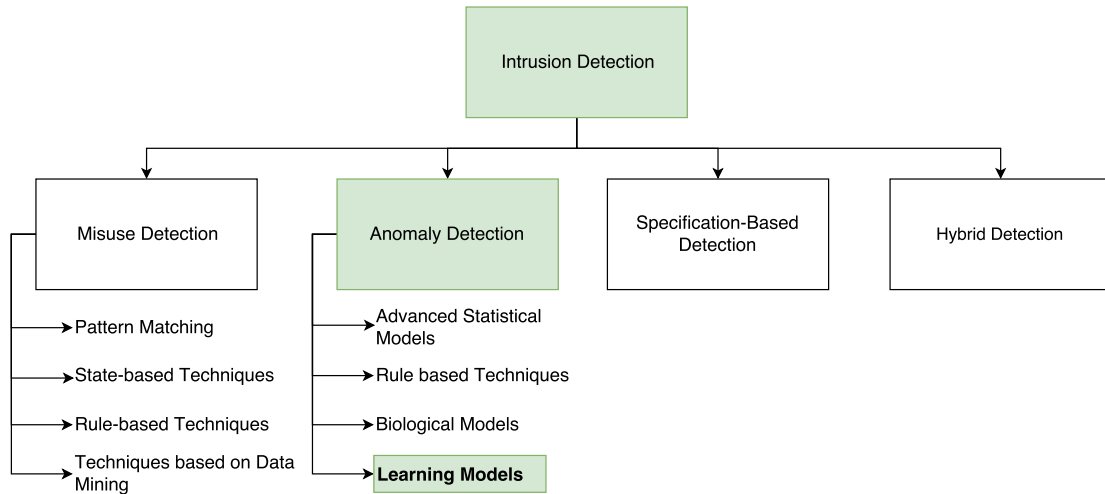


FIGURE 1.1: Intrusion Detection Approaches

Misuse detection is highly effective in detecting known attacks. However it is insufficient against unknown or novel attacks because their signatures are not yet known. Additionally, any modification to the signatures can cause an increase in false alarm rate and that will decrease effectiveness and reliability of the detection system. Specific-based detection aims to set particular behaviour based on the default deny principles. If the specifications are violated, the system will think there is an abnormal situation. This approach is effective to uncover the unseen attack that may be carried out in the future. However, setting particular specifications to the system is an overwhelming task in considering each different problem. Anomaly based detection approach is, basically, constructed on normal activity profile and it assumes that any adversary action will conflict with the normal activity. Anomaly based detection is examined under four subheadings; advanced statistical models, rule-based techniques, biological models and learning models. We adopted learning models, because of that even if a misuse detection can give faster response. Learning models, have a more robust structure against unknown attacks than others [10].

Traditional ML methods, such as Bayesian Belief Networks (BBN) [11], [12], Support Vector Machines (SVM) [13], [14], [15], [16] and others [17], [18], [19], [20] have been applied for cyber security, however the large scale data generation in IoT calls for a deep learning based method which performs better with large data sizes and is adaptable to different attack scenarios.

IPv6 is a commonly used protocol in IoT and IPv6 based wireless sensor networks (WSNs) [21] are particularly susceptible to routing attacks. IPv6 over Low-powered Wireless Personal Area Networks (6LoWPAN) is an IPv6 based WSN protocol [22]. 6LoWPAN presents some advantages as low power consumption, tiny, small foot print, inexpensive structure and easy maintenance [23]. Besides, WSNs include many sensors which have limited resources such as low memory, small bandwidth and low energy. Locations of routing attacks in IoT are also shown in Figure 1.2.

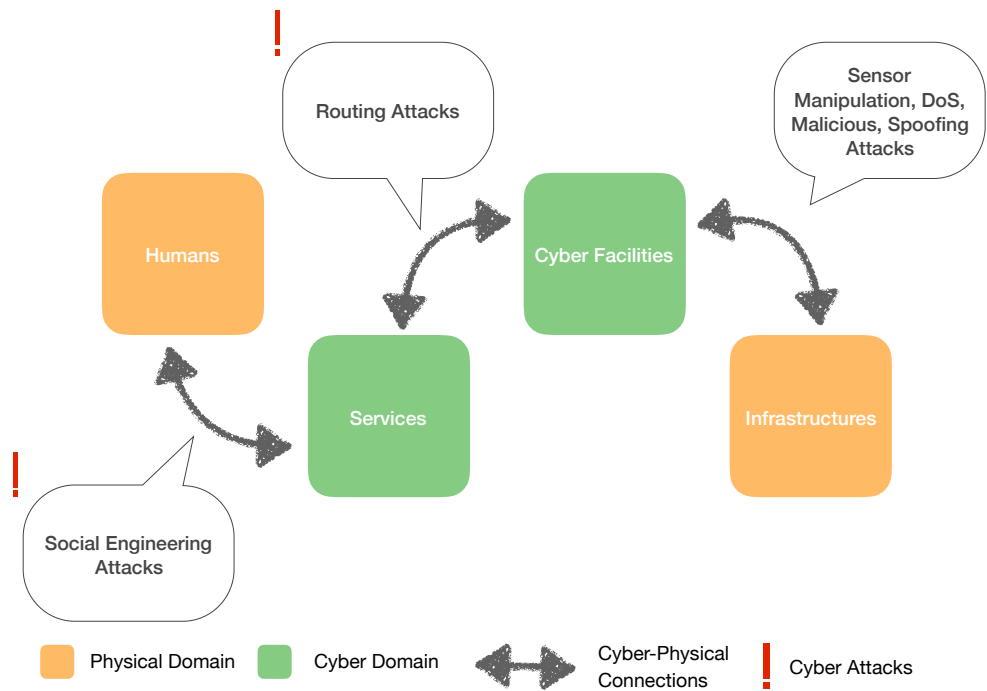


FIGURE 1.2: An overview of IoT attacks

In our study we have demonstrated the viability of our methodology with simulations up to 1000 nodes whereas the existing studies, such as [24], [25] and [26], have demonstrated their methodology with little number of nodes (up to 50), which is not a realistic approach for an IoT environment. We used data generated by real-life equivalent simulations because of a lack of availability of public IoT attack datasets. The Cooja simulation generates raw packet capture files, which are first converted into Comma Separated Values(CSV) files for text-based processing. The CSV files are then input to the feature pre-processing module of our system. The features are calculated based on the traffic flow information in the CSV files. First, feature conversation process is applied to some features, which is located in raw datasets, then, we have identified 12 features as an

initial candidate feature set. Afterwards, feature normalization is applied to all datasets to reduce the negative effects of marginal values. In the pre-feature selection step, we have analyzed the importance of features by Randomised Decision Trees [27], histograms and pearson [28] rate calculation. As a result of this analysis, some of the features are dropped in pre-feature selection process. After feature preprocessing, the datasets corresponding to each scenario is labelled and mixed to produce a preprocessed dataset, consisting of a mixture of attack and benign data. These datasets are fed into the deep learning algorithm. Deep layers are trained with regularization and dropout mechanisms, their weights are adjusted and the IoT Attack Detection Models are created. This methodology is also depicted in Figure 1.3.

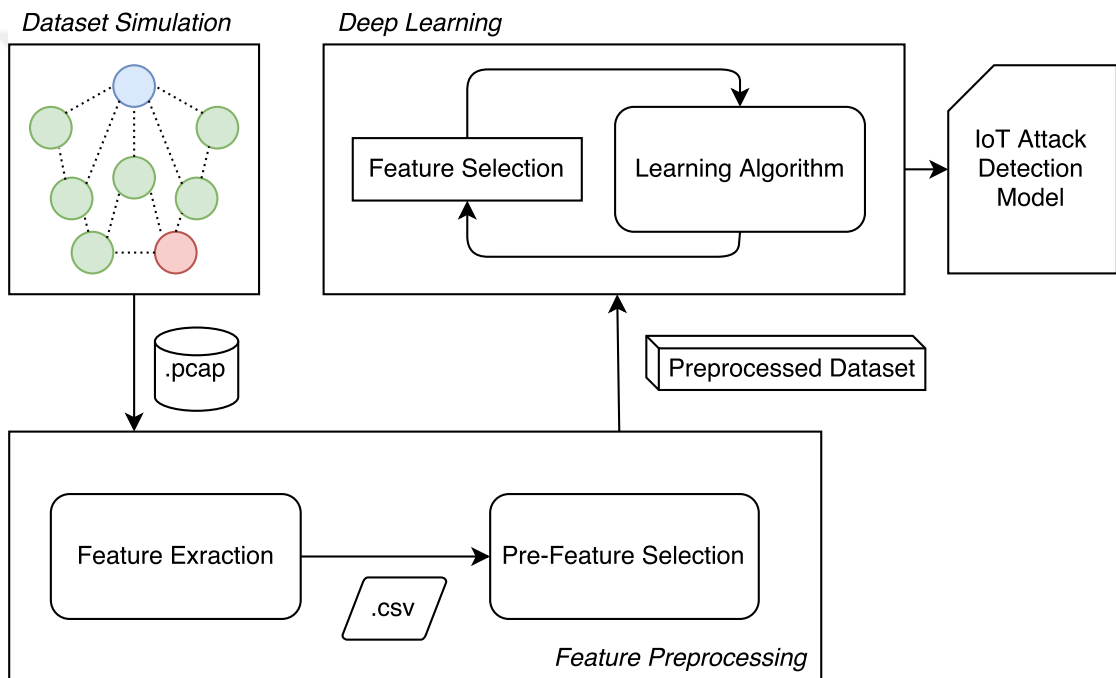


FIGURE 1.3: Methodology Flow Diagram

IoT devices are resource constrained and have power consumption limitations. As a result of these constraints the security mechanisms devised for IoT should be efficient and lightweight, putting as little computation and communication burden on the end-devices as possible. Our proposal puts minimum burden on the IoT network since it requires only the network packet traces for detection and prediction of attacks which can be collected externally by a network recording equipment or specially designated nodes. According to our knowledge we are the first to prepare a deep learning based methodology for routing attack detection.

The objectives of this study are:

- To deeply understand the IoT, threats, risks and routing attacks.
- To create routing attack datasets and their preparation.
- To build a neural network by deep learning and train them by produced datasets.
- To evaluate the models.

1.4 Contributions

We produced three IoT dataset that include 3 different routing attacks. Then we trained the neural networks with the preprocessed datasets. After that, we got 3 IoT attack detection models for each routing attack. The models are also generalisable by taking into account different network topologies. The training metrics of these models are listed in Table 6.1 as Training Accuracy (up to 99.5%), Training Loss (max. 5%) and the performance of these models are listed in Table 6.5 as F1-Score (up to 99%).

1.5 Outline

This thesis is organized into seven chapters. The first chapter provides brief introduction to thesis and explanation of its methodology. Then:

Chapter 2 explained background and discuss about IoT, its fundamentals, challenges, threats and risks. Routing attacks to IoT are explained in detail.

Chapter 3 give a literature review on detecting routing attacks by using different approaches.

Chapter 4 explained the whole dataset generation process, step by step.

Chapter 5 focused on Deep Learning Based Detection of Routing Attacks.

Chapter 6 is evaluation section, performance, analysis of project and discuss differences and contributions of this thesis and make conclusion.

Appendix includes scripts that we implemented in the thesis.

Chapter 2

Background

2.1 Internet of Things (IoT)

Internet of Things (IoT) is one of the biggest innovations of the this century, considering the impact on our daily life. The areas of its usage are rapidly increasing. In 2017, number of devices, that are called IoT, is approximately 27 billion and this number is estimated to reach 50.1 billion by 2020 [29]. In another aspect, market size of IoT is estimated to reach approximately \$9 trillion by 2020 [30]. Basically, IoT is a network that contains softwares, nodes and servers.

Before the IoT, CPS's should be mentioned to comprehend IoT. Because, even if the term of CPS and IoT have close meanings, CPS includes IoT.

2.1.1 Cyber-Physical Systems (CPS)

CPS have sensitive data, ensure collaborative business, produce high efficient energy and enhance life's quality. CPS are smart systems which provides an environment for cooperation of computational components and things that are well-known with their physical activities. CPS is kind of a bridge that brings cyber and physical domains together and takes upon itself an indispensable responsibilities in many areas as clearly depicted in Figure 2.1. For instance, health sector recently goes towards to use CPS for getting more information of patients and serving them effectively [31].

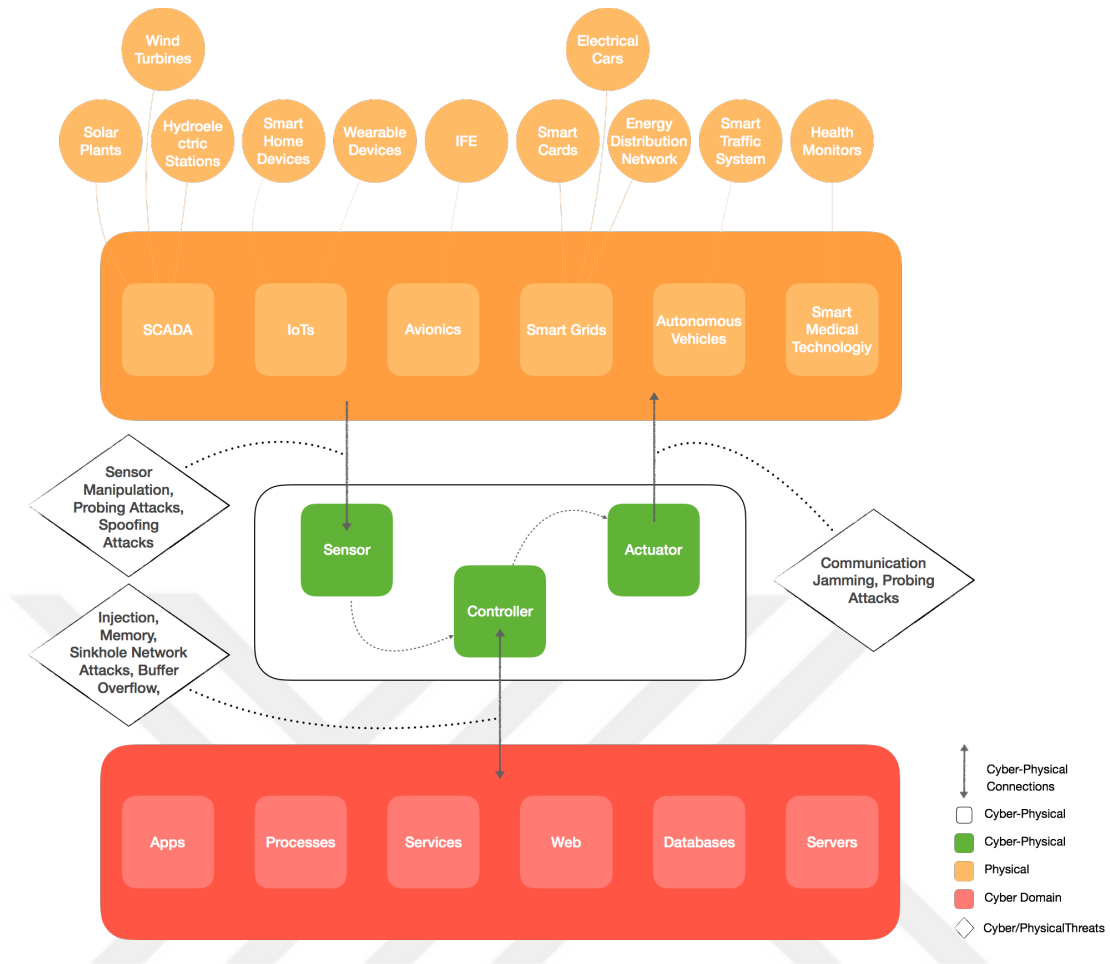


FIGURE 2.1: CPS Overview

Sensor, controller and actuator are basic components of CPS that can sense the environment with sensors, make a decision with controllers and force the physical domain to execute the decision with actuators. They are shown in Figure 2.2.

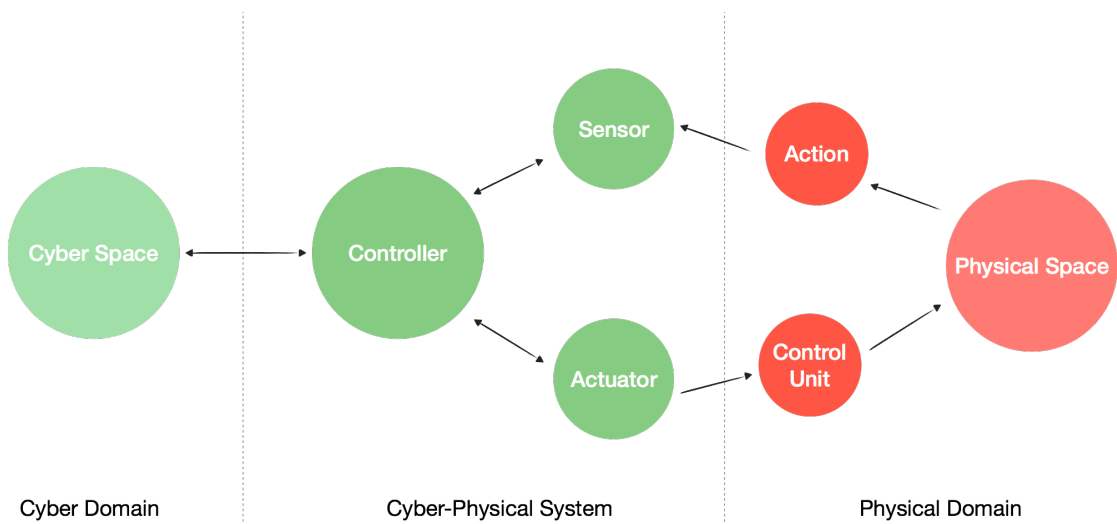


FIGURE 2.2: Components of a CPS

Heterogeneity and complexity [32] make CPS are susceptible to physical and cyber attacks. For instance, spoofing or denial of service attack violate the supervisory control and data acquisition (SCADA) systems [33]. The attacks to CPS became more of an issue, because of the importance of CPS [34].

2.1.2 IoT

IoT is identified as a network of distributed devices that interconnected with softwares, servers, sensors and etc. If a 'thing' is mentioned as a part of IoT, it can be reachable, locatable, addressable within the cyber world [35]. This features makes the devices more usable and serviceable. The contributions affect the living standard positively. The reason of market size and usage values of IoT, which are mentioned above, are clearly understood at this point. People, business and governments prefer this technology for the ease of reachability it offers.

The list of IoT usage areas is way too long, to name a few; smart home and it's devices, wireless sensors, smart locks, smart meters, wearable devices, security cameras, smart plugs, Radio Frequency Identification (RFID), Machine to Machine(M2M), Machine to Human(M2H) devices and so on. The usage values and areas revealed that the IoT touches (or will touch) every point of human life.

2.1.3 Challenges of IoT

IoT devices have the ability of data collecting, transferring and processing in smart applications [36]. This data types spread to many areas such as health, transportation, military etc. Security of the sensitive data, thats the biggest risk of IoT, comes into prominence. This risk roots in two main vulnerabilities. First; heterogeneous devices and inter-operable connections makes the management of IoT systems more complex. Second; many devices have resource limitations, lack of computational capability, low latency. Second reason also makes the detection of possible and unknown attacks to IoT devices difficult [37]. These reasons make the routing protocols vulnerable. For example, WSN consist of nodes that include one or more sensors which have low-cost and limited power. These sensors' objectives are sensing the environment and communicating

with the base station. The base station has more energy, communication and computational power than other nodes to assure the network between other nodes and end user. These abilities ensure monitoring environmental conditions by nodes [38]. WSN has general characteristics such as, having small number of nodes, power limited nodes, dynamic network topology and large scale of deployment [21], [39], [40], [41]. WSN has important issues as sensitive data protection, privacy and authentication. Unfortunately, the conventional solutions to these issues, as cryptography or key distribution protocols, couldn't be applicable due the mentioned constraints of sensor devices[42]. There are many different routing protocols in WSN. Some of them built on resource awareness or energy reduction, but none of them are robust routing protocols against the IoT attacks.

2.1.4 Threats and Risks of IoT

Understanding threats and risks of IoT is an introduction to understanding the attacks to IoT. For this purpose, the architecture of IoT should be examined. In the Figure 2.3, WSN and WSN Border Router stacks are depicted to understand how the data are conveyed to the server, cloud or database.

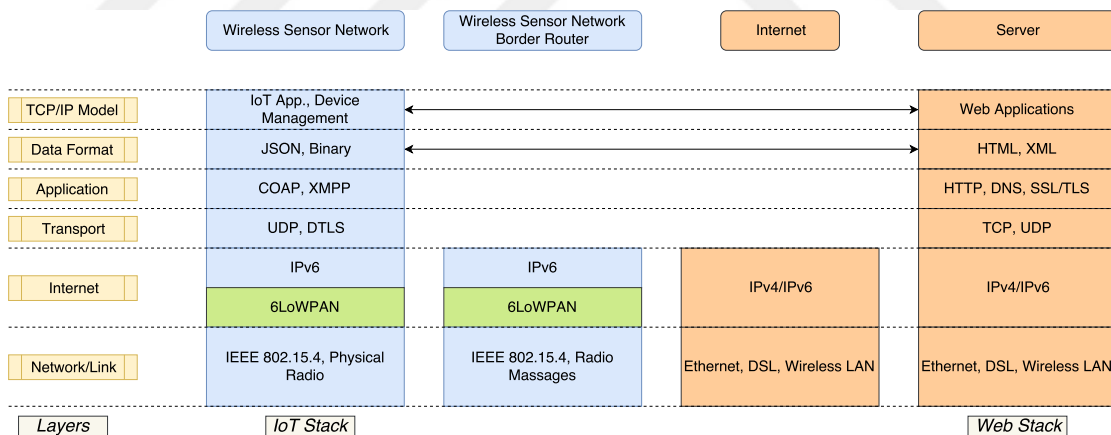


FIGURE 2.3: IoT Architecture

The trust between nodes are assumed by most WSN protocols because the authentication between the nodes brings communication overhead. This assumption unsurprisingly creates a risk that enables the malicious nodes to easily manipulate the network [43].

IPv6 is an important root of IoT, like an enabler. Because IPv4 cannot afford the size of IoT systems. Security suggestions and considerations of IPv6 are the basis of IoT security [44]. IoT has the same threats with IPv4. Additionally, IoT are subject of the

unprecedented threats due to its location that is at a junction point of cyber domain and physical domain. Briefly, the expanding attack surface is a threat. An attack could manipulate the information and that can cause the unintended action in physical domain. In Figure 2.4, system of IoT is clearly explained from sensors to data analysis stage.

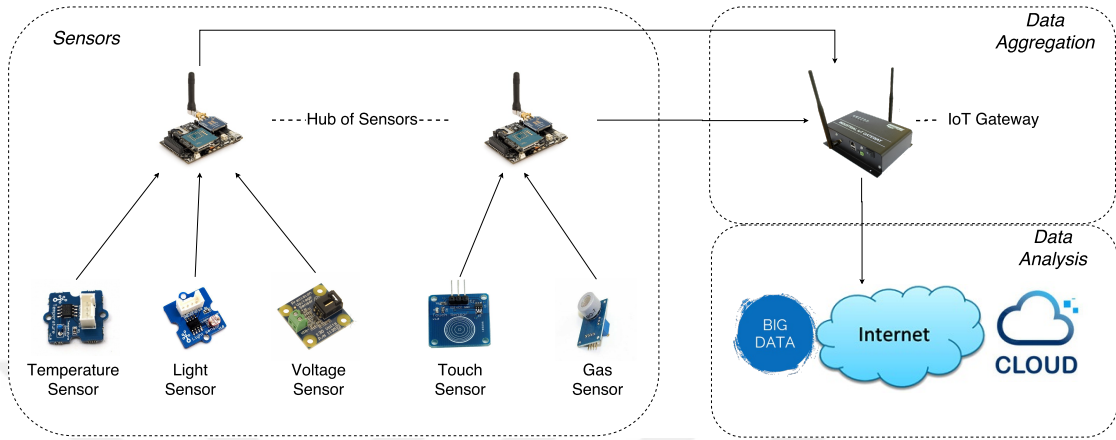


FIGURE 2.4: System of IoT

There are many type of attacks to IoT that are physical attacks, reconnaissance attacks, DoS, access attacks, attacks on privacy, cyber crimes, destructive attacks and SCADA [45].

Routing attacks are performed at network layer and they are more critical than other attacks, in other saying, they can be an initializer for rest of the attacks to IoT.

2.1.5 Simulation of Routing Attacks

We used the Cooja IoT simulator to simulate different IoT network communication scenarios. Cooja, coupled with the Contiki operating system, is a cross-level (application, operating system and machine code layer) simulation tool [2]. Sensors in the simulated network run with the Contiki operating system and implement the Routing Protocol for Low-Power and Lossy Networks(RPL) protocol. Contiki makes possible to load and unload individual programs and services to the simulated sensors [46]. We have conducted a simulation of each attack as mentioned above, by running real sensor code in Cooja simulator. We made the simulations on cloud based system. The contiki environment includes 64-bit Java Runtime Environment on top at 64-bit Ubuntu operating system and contiki 3.0. Cooja user interface is shown in Figure 2.5.

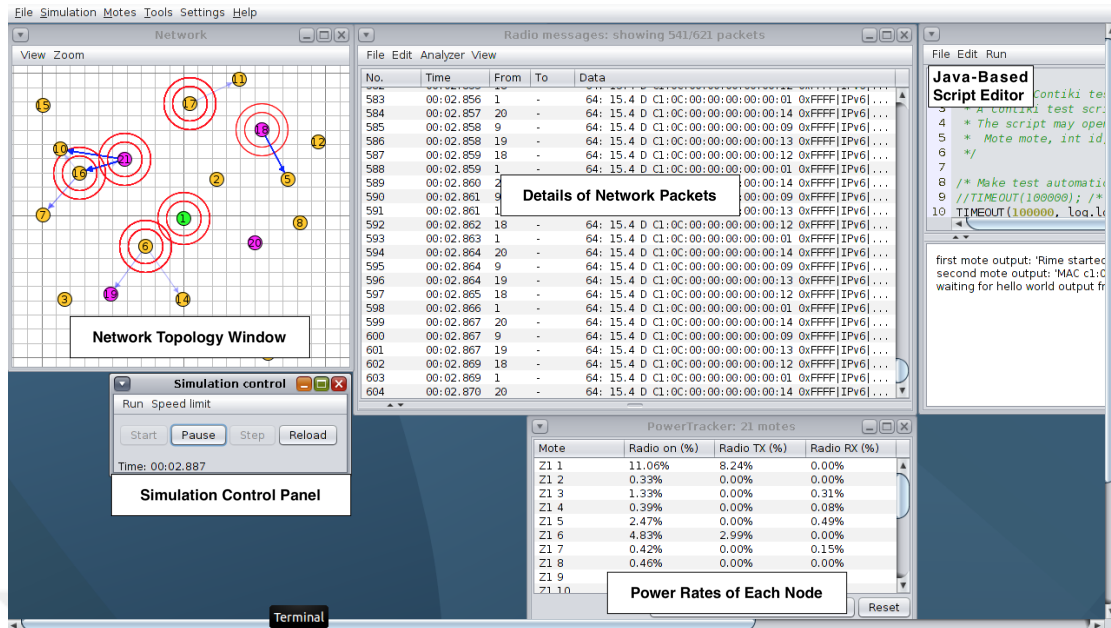


FIGURE 2.5: Cooja User Interface

We have generated various attack scenarios at a wide scale number of IoT nodes, ranging up to 1000, with different percentages (5%, 10% etc.) of malicious nodes for simulating routing attacks such as decreased rank, hello-flood and version number attack. Scenarios are listed in Table 4.1. We simulated these scenarios by the Cooja network simulation, avoiding to produce a synthetic dataset, since Cooja enables to run actual RPL code on the simulated nodes. A sample of IoT simulation are shown in Figure 2.6. Cooja also enables to take results of radio messages of the simulated networks as a PCAP files.

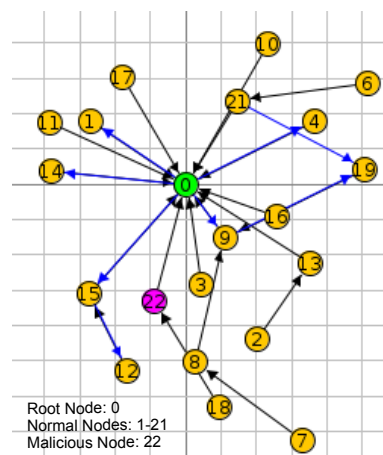


FIGURE 2.6: A sample of the Simulation

2.1.5.1 Routing Attacks to IoT

IoT applications are located in many fields such as smart homes, smart energy monitoring, healthcare systems, smart cities, logistics and etc. Because of this wide range usage, security of IoT is important and routing attacks are a very common threat for IoT [22].

RPL [47] is a kind of distance based protocol. First, each node in the network determines its routing path, then RPL network initializes. In another aspects RPL is a tree-oriented IPv6 routing protocol for 6LoWPAN and it creates Destination Oriented Directed Acyclic Graphs (DODAGs), called as DODAG tree. Each network has one or more DODAG root node as central node and each network has a unique identifier DODAG ID to be identified. Additionally, each node has a rank number and a routing table due to the other nodes' rank numbers. The rank number is used to determine the distance between the node and root [48]. An example of RPL network is depicted in Figure 2.7.

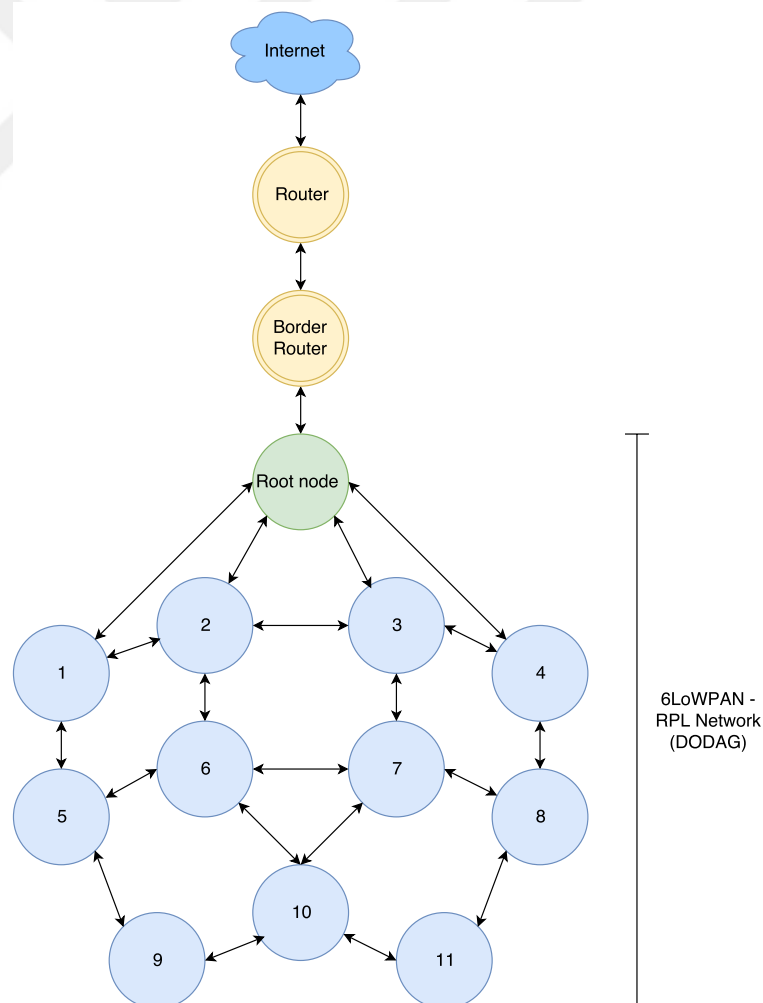


FIGURE 2.7: Sample 6LoWPAN Concept

In the RPL, there is three type of control packets; DODAG Information Object (DIO), Destination Advertisement Object (DAO) and DODAG Information Solicitation (DIS). DIO packets are firstly sent by base (or root) node as broadcasting to create DODAG tree. The rest of nodes receive this DIO packets and they create their routing table by selecting parent node. They send DAO packets to parent node to ask permission to connect to the parent node. Parent node accepts this offer by sending a DIO ACK packet back. New node sends DIS packets to join DODAG tree. If a new node joins the tree, all nodes send DIO packets again to reform DODAG (or network topology).

RPL attacks can be examined under three categories depending on the vulnerability which they aim to exploit. These categories are resource-based, topology based and traffic based. Resource-based attacks aim to consume energy, power and overload the memory. Topology-based attacks aim to hinder the normal process of the network. This could cause that one or more nodes are broken off from the network. Additionally these attacks threaten the original topology of the network. Traffic-based attacker nodes aim to join the network as a normal node. Then these attackers use the information of the network traffic to conduct the attack [49].

Routing attacks take place at the network layer. IoT systems are generally vulnerable against routing attacks. Among the most significant routing attacks are decreased rank, hello-flood and version number attacks. Location of routing attacks in IoT systems also depicted in Figure 2.8.

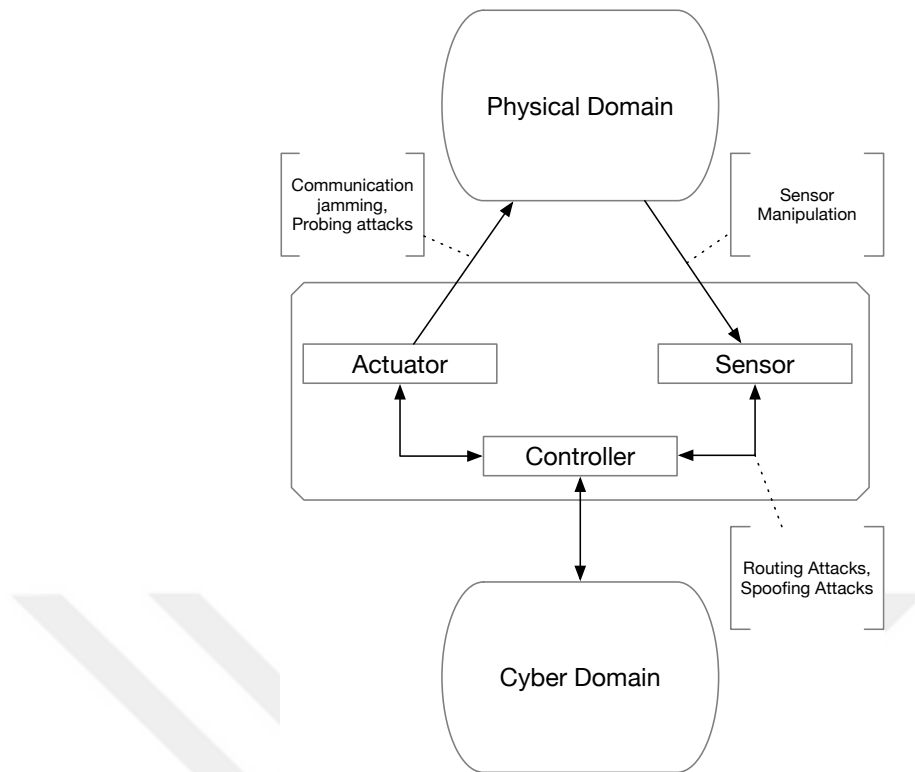


FIGURE 2.8: Routing Attacks in IoT

The decreased rank attack is such a traffic misappropriation attack. In decreased rank attacks, malicious nodes advertise lower rank of other nodes to neighbour nodes by sending DIO packets. So the neighbour nodes change their routing path including the attacker node by sending DAO packets. Decreased rank attack can be applied to make an introduction for blackhole, eavesdropping and sinkhole attacks. Decreased rank attack is also visualized in Figure 2.9. In this figure, node 1 is the DODAG root node and the others are normal nodes except for node 9, which is the malicious node that conducts the decreased rank attack in the network. The nodes, 3 to 8, are not effected by the attack. Nodes 10 and 11 are partially affected from the attack and their communication is partly interrupted. Some of the packets transmitted over these nodes could be taken by malicious node because the malicious node is in their routing table, in other words, they can send some packets over malicious node to convey coming packets to destination. The nodes, 12 to 18, are the victim nodes whose entire communication is transmitted over malicious node.

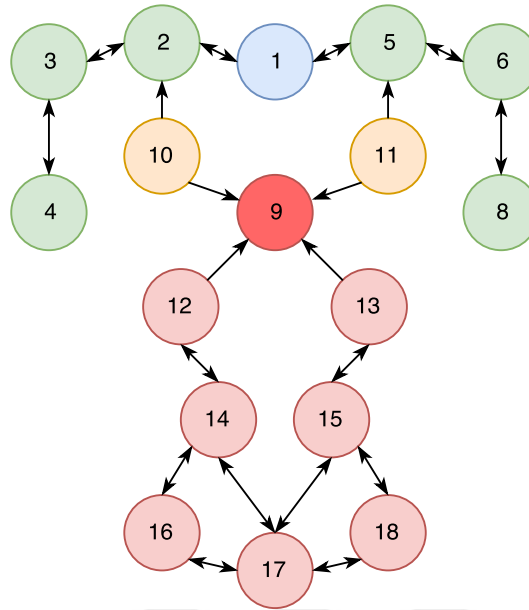


FIGURE 2.9: Decreased Rank Attack

It's clearly seen that number of received packet by the malicious node increases when attack happens. We aim to use this anomaly as features. So we extracted Reception Rate (RR) (2.1), Reception Average Time (RAT) (2.2), Received Packets Counts (RCP), Total Reception Time(TRT). Additionally, DIO and DAO packet count are calculated because of that the attack is started by DIO and DAO packets.

$$RR = \frac{\text{Received Packet Count of the Node}}{1000[ms]} \quad (2.1)$$

$$RAT = \frac{\text{Total Reception Time}}{\text{Received Packet Count of the Node}} \quad (2.2)$$

In equation 2.1 and TR , 1000 is millisecond within all simulation. We have applied windowing while extracting features. The windowing process will be explained in Chapter 4, section *Feature Extraction*.

The main purpose of the HELLO message is to introduce and integrate new nodes to the network. The nodes broadcast HELLO messages with their own metrics such as signal power and ID number. All the other nodes create their own routing table to send their messages. A malicious node sends HELLO messages by DIS packets to his victims by strong signal power and suitable routing metrics, appearing like an neighbour node.

The attacker node becomes the most favorable for the victims. This attack is called the hello-flood attack. The initialization part of this attack is depicted in Figure 2.10.

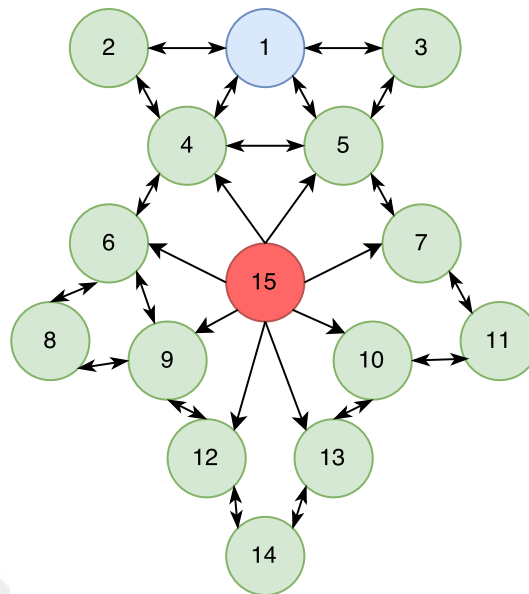


FIGURE 2.10: Before Hello-flood Attack

The malicious node, Node 15, broadcasts HELLO messages to Nodes 4-13, except Nodes 8 and 11. The victim nodes change their routing table because the malicious nodes advertise high quality metrics. After that, effect of hello-flood attack is depicted in Figure 2.11.

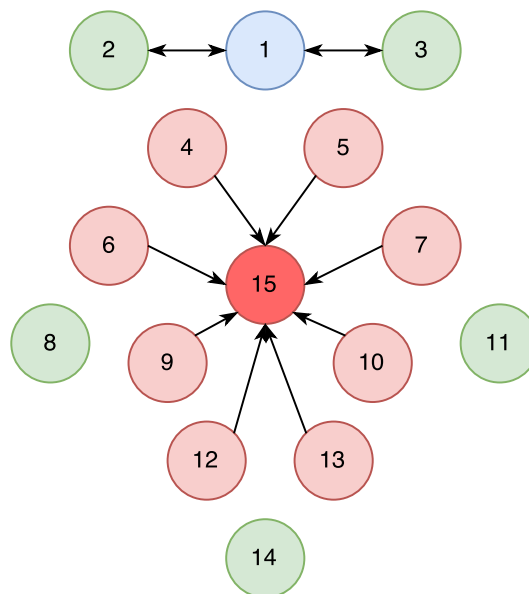


FIGURE 2.11: After Hello-flood Attack

It's obvious that number of transmitted packets of malicious node increase. So we extracted Transmission Rate (TR) (4.2), Transmission Average Time (TAT) (2.4), Transmitted Packets Counts (TPC), Total Transmission Time(TTT) and DIS features to identify this attack.

$$TR = \frac{\text{Transmitted Packet Count of the Node}}{1000[ms]} \quad (2.3)$$

$$TAT = \frac{\text{Total Transmission Time}}{\text{Transmitted Packet Count of the Node}} \quad (2.4)$$

In RPL, version numbers of nodes are changed by root node. When the root node changes them, each node starts the communication for reconstructing of their routing table. So the network topology is changed. In version number attack, malicious the node changes its version, then other nodes are forced to change their routing table. So the malicious node can promote itself to take better a place in the other nodes' routing tables. This can jeopardize the network's information security and performance due to the change in topology [50].

2.2 Deep Learning for Cyber Security

We investigate the use of deep learning (DL) for detecting routing attacks that target to IoT. Before giving information about DL, ML should be explained to better understand DL. Because, ML can be seen as the ancestor of DL.

2.2.1 Machine Learning

Machine Learning(ML) is on of the pathways in leading Artificial Intelligence(AI) research. Popularity of ML comes from two purpose or two task have to be done by ML. First the task that can be done by machines, second the task that can't be performed by humans. Learning activity comes into prominence to be intelligent what refers a system has ability keeping up with changes of it's environment. If a system can accord to the changes, this ability can help it to survive. ML studies are placed at the junction area of statistics, computer engineering(CE) and computer science(CS). But it can also provide

solutions to other disciplines. Because application of ML depends on using data that can be from finance, geoscience or archaeology. ML also produce information from the data. Such that computers use data while working on a process but the data, usually, is meaningless from human perspective. We can solve deterministic problems easily with CE. For example, a software to control lighting system in a smart home can do a good job all the time, using the activity and daylight. But there are lots of non-deterministic problems that we don't have enough information about or power and time to solve. We need statistics for solving these kind of problems. For instance, modeling the fear of humans against unexpected situation is so hard without statistics. Basically, statistic builds mathematical models and ML helps to train them.

From this point of view, ML has very similar meaning with classical programming. This description make easier to see the difference between classical programming and ML; a classical program is fed by data and rule as an input which transform the answer as an output after the process, in the contrast, a ML algorithm is fed by data and answer, which is expected to understand the relationship between them. This relationship can be used in different forms of data to estimate their outputs.

In ML, there isn't any explicit programming, it should be called 'training' because of the learning process. ML system does not do anything except for exploring the statistical structure of given data. As an example, which is relevant with the next task, the real data. For example, someone, who let's say is our doubting Thomas, wants to know the cause of a certain activity that happens in the night-time around his home. He puts surveillance cameras to certain points of his home and one of the important features of the cameras is taking photos and saving them on the computer when they detect any activity with their sensors. But the security system of the home is very tedious and boring, because the owner of the home has to check all the photos, of last night, every day to be satisfied about his home's safety. If we wish to establish a better system, it has to warn Thomas when a real threat occurs. The system shouldn't give an alarm when the night activity happens because of a cat or squirrel. So, We can give lots of examples of pictures to ML algorithm, that are already tagged before, called target variable, before. All this target variables include the features. Example pictures, called training set, include objects of humans, cats, leafs etc. Then, the ML algorithm learns statistical rules

for correlating definite picture to definite tags and the output is an ML model that aims to detect night activity. Finally, the system analyzes all night activity for each day over taken pictures and if there is a suspicious activity of interest, it gives a notification or an alert, as our doubting Thomas wanted from the beginning. Some researchers study on ML process in two parts; the learning part and the inferring part. As mentioned above, the learning part is feeding the ML algorithm with the training set and the inferring part is making predictions about the cause of activity by the system.

Supervised learning and unsupervised learning are main types of ML, first of them are exist by using fully labelled dataset whereas other one are exist fully unlabelled dataset [51]. In supervised learning, the model receives datasets which includes some features vectors and labels that are the corresponding outputs of feature vectors. Thus the model learns to produce correct outputs as a result of a given new input. Classification and regression are most popular product of supervised learning [52].

In unsupervised learning, the other way round, there is no supervisor who supplies the labels, that include correct result of corresponding input, to train the models. So the model, has only input values, observes the results of its action. In the other referring, Unsupervised learning is an enterprise to describe hidden patterns from input data. Clustering and dimensionality reduction are two common unsupervised learning example.

2.2.2 Deep Learning

Deep learning (DL) is a kind of Neural Networks(NN) training and has NN architecture. Difference between 'old school' NN and deep learning is that DL has many hidden layers [52]. DL also learns the features itself, which enables the learning process to be more accurate and also it is shown to be more efficient and accurate than shallow learning [53].

DL has reached success in the computer vision, pattern recognition, image and audio processing. It also enabled significant improvement for classification and prediction problems [54]. Complex deep neural algorithms are trained with the use of powerful GPUs. For

AI, DL represent the state of the art, also for dealing with Big Data especially regarding scalability and generalization.

In supervised learning, there are three type of datasets. First, training set is one of the key terms of learning process. It is an enabler for learning algorithm to be supervised and it contains the expected results under the label feature. The weights of the internal layers of NN are determined in light of the data and their expected results. And optimal weights are obtained. Another term is validation set, that assists the learning process for tuning the parameters of functions to get optimal weights. Finally, the test set is used for evaluating the performance of training process. Before starting the learning process, the dataset is separated into the training set and the testing set, that validation dataset is split from the training set. When preparing the neural network algorithm, epochs are used as learning time. One epoch means, the training set is passed through the network completely. NN training algorithms aim to determine the 'best' possible set of weight values for the problem under consideration. As expected, determining the optimal set of weight is often a trade-off between minimizing the network error, computation time and maintaining the network's ability to generalize.

Neurons are the main actor of the learning section. They take one or more input from the previous neurons with the connection weights, sum them up, put it in the activation function and produce an output (2.6) that is, basically, fired or not. Mathematical representation of the neuron is shown in 2.5. After this addition, the activation function puts the Y into process.

$$Y = \sum(input) * (weight) + bias \quad (2.5)$$

$$Output = f(Y) \quad (2.6)$$

'Fire' means to activate, the name is inspired from the biological working of the brain. The similarity between neurons (or brain cells) and artificial neurons is depicted in Figure 2.12. For the brain cell, electrical signals from other cells are transported to the cell body by dendrites, then output electrical signals are forwarded along the axon to other brain cells. Working logic of artificial neurons, are explained above, has similar a process.

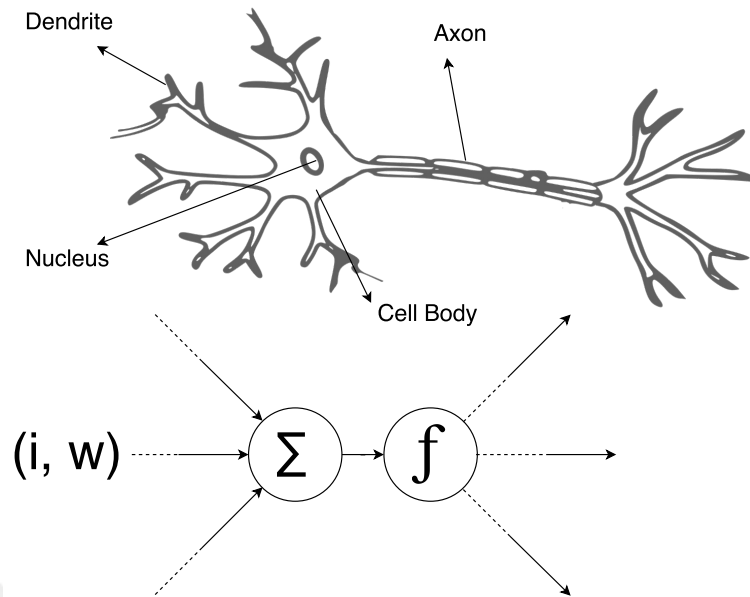


FIGURE 2.12: Brain Cell and Artificial Neuron

Step function is an activation function that takes Y ; if Y is above a certain value (or threshold), the output of step function is activated, otherwise the output is non-activated. Step function graph is showed in Figure 2.13. In this figure, x is threshold.

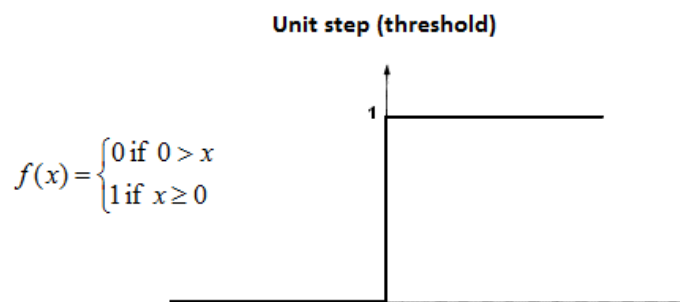


FIGURE 2.13: Step Function

Sigmoid function is like a smooth version of step function. Differences are non-linearity and better classification result. Sigmoid function graph is showed in Figure 2.14.

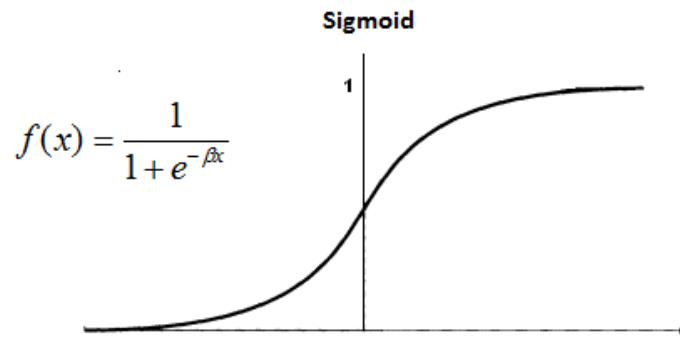


FIGURE 2.14: Sigmoid Function

Tanh function has quite similar function with sigmoid function, it has boundaries $(-1,1)$.

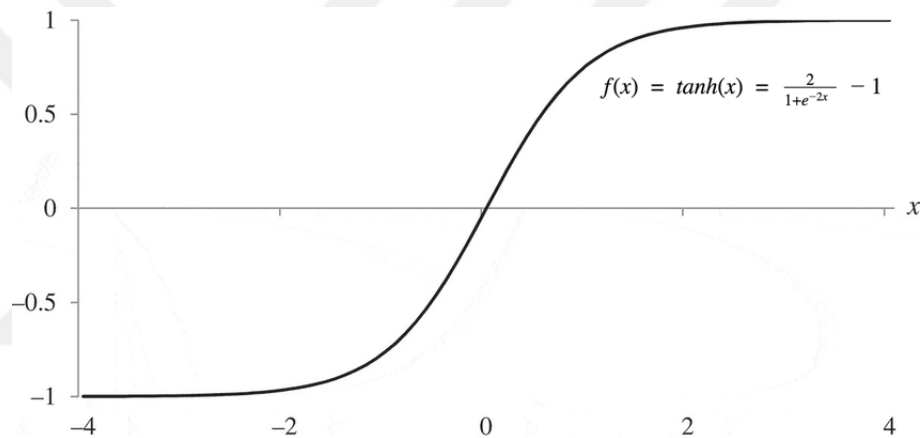


FIGURE 2.15: Tanh Function

Rectified Linear Unit (ReLU) is the function that is most effective solution for vanishing gradient problem in our study. The mathematical equation is shown at 2.7.

$$y = f(x) = \text{maximum}(0, x) \quad (2.7)$$

Output is zero for the input that is less than zero, whereas output is equivalent to input that is more than zero. For binary classification, ReLU function is more suitable and we use it in the hidden layers as activation function. ReLU function is also showed as graphical representation in Figure 2.16.

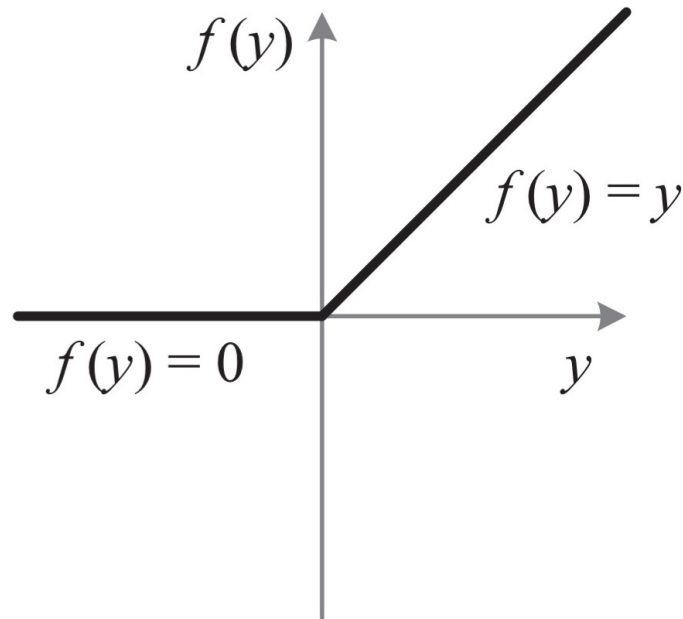


FIGURE 2.16: ReLU Function

NN layers include neurons. Input layer is the first layer of NN, there should be same number of neurons like number of feature or size of dataset. Output layer is the last layer of NN. For binary classification problem, there should be one neuron with activation function. NN have also different type of layers, named hidden layer, that are located between input layer and output layer. Number of hidden layers make NN deep, so, they are called Deep Neural Network(DNN) and learning process in the DNN is DL. Number of neurons of hidden layers should be like a triangle according to best practices. For example, there is three hidden layers and number of neurons might be 100, 200, 100, respectively. The structure of our deep neural networks can be another example (Chapter 5, Figure 5.1).

Dropout rate (or dropout regularization) is another parameter in DL. Randomly selected neurons, at the rate of dropout rate, in hidden layers are neglected during the learning process that means their effect to weight update and activation functions. Briefly, randomly selected neurons are dropped-out. It helps to decrease the sensitivity to particular weights and supports learning process to avoid the overfitting problem [55]. The overfitting problem, in a few words, is that the noises and details in training data that are focused by the learning model and it learns the training model way too much. If the learning model are tested with another dataset of same problem, testing results will be quite a change.

The weights of neurons within each layer are tuned by applying back-propagation formula. Vanishing gradient problem occurs when lack of applied back-propagation formula.

$$\Delta W_t = \alpha * \frac{\partial MSE}{\partial W_t} + \mu * W_{t-1} \quad (2.8)$$

The formula of back-propagation is shown in Equation 2.8; W is change in the edge weight of the time t (or $t-1$ means previous iteration), $alpha$ is learning rate and derivation within fraction is the gradient.

While the training, loss functions are used for calculate the loss between expected variable and output and they are very helpful to train a neural network. For example, mean squared error(MSE) is most popular loss function in the learning study, particularly, ML/DL beginners are prefer to use it. The function of MSE is shown at 2.9. y_i is the output of learning process when y_i with accent mark is expected result and n is number of output classes.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (2.9)$$

Cross entropy loss(CEL) is another popular loss function and commonly preferred for classification or regression problems. CEL is used for determining maximum likelihood in statistics and it has better results than MSE. Function of CEL is shown at 2.10. i is the number of training instance, y_i with accent mark is expected result and y_i is the output of learning process [56]. MSE and CEL are widely used in classification problems.

$$CrossEntropy = \sum_K^{i=1} y_i \log(\hat{y}_i) \quad (2.10)$$

Chapter 3

Related Work

Garofalo *et al.* present, in their paper [57], an implementation of Intrusion Detection System (IDS) for WSN. This solution addresses a kind of mesh network. The paper improves the architecture that was presented with recent studies which presented a hybrid structure, that are processed by Centre Agent(CA) and Local Agent(LA) for intrusion detection. Information about implementation of CA isn't detailed in the recent studies, so this paper also takes in account this topic. In the presented implementation, CA makes detection via a decision tree (DT) . LA monitors the gathered data from all nodes and makes the detection. If there is an attack, LA sends a security alert to CA and the node, which is source of detected attack, is taken into a blacklist. Then the node isn't used anymore for further data packages. Additionally, issues and presented solutions in recent studies are compared on intrusion detection rate, false positive rate, low power consumption rate and anomaly detection rate. The dataset is created in a simulated environment, with ns-3 [3]. Different nodes are sources of data packages and the data traffic between the nodes is captured for 4 hours. The Ad hoc On demand Distance Vector (AODV) protocol [57] was used as routing protocol. Specifically, the dataset involves the sinkhole attack, known as a routing attack. This sinkhole attack is produced on AODV in the simulated environment.

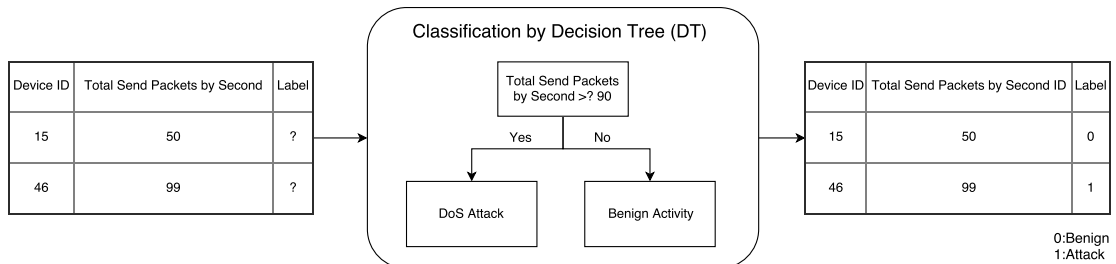


FIGURE 3.1: Decision Tree's Logic

There are many attacks, that aim to exploit WSNs, from the network layer. Detecting and identifying the attacks analysed in [58]. Attacker tries to prevent the network traffic that is passing over the base node. This way, all WSN's traffic is open to the attacker. The attacker impersonates an ordinary node, sends self routing metrics, that are designed to be better than others, to his neighbour nodes. After that, neighbour nodes compare the new metrics to their own metrics. Attacked neighbour nodes replaces their routing metrics with the new one because of the attackers routing metrics are better than theirs. Thus the attacker aims to be the preferred node by other nodes. Two different communication protocols are also explained in the paper. MintRoute is the most preferred routing protocol in TinyOS, is a small developed operating system for WSN. MintRoute isn't a complex protocol when it is compared the other protocols and it is also easy to work with sensors in WSN. MintRoute protocol decides to select best route, to send packets based on link quality. Link quality means signal strength, packet loss and end to end delay [59]. Link quality is updated at certain intervals by 'Route Update' packets and each node calculates the their neighbour's link quality due to the 'Route Update' packets. This calculations are recorded with neighbour's ID in memory. Then these link qualities are broadcasted periodically. Finally, each node selects their 'parent node' from the link quality in their memory and changes their 'parent node' in two conditions; 'parent node's link quality drops under 25% or another neighbour's link quality becomes 75% better than 'parent node' [60]. TinyAODV is a lighter version of AODV protocol. When the source node sends a Route Request (RREQ) packet to select best route for itself. Adversary node replies a Route Reply (RREP) packet to the source node immediately, instead of broadcasting it to his own neighbours. Attacker's RREP packet includes information of path with lower number of hops to the destination node. So source node chooses the attacker node as a parent and sends its packets over to the attacker's. In the paper, detecting sinkhole attacks are briefly explained. Because what

if compromised node is located next to the base station, all the WSN, instead of a few nodes, is under the threat of the sinkhole attack. The sinkhole attack takes part in the routing process and this ensures that it can launch new attacks as selectively forwarding dropping packets.

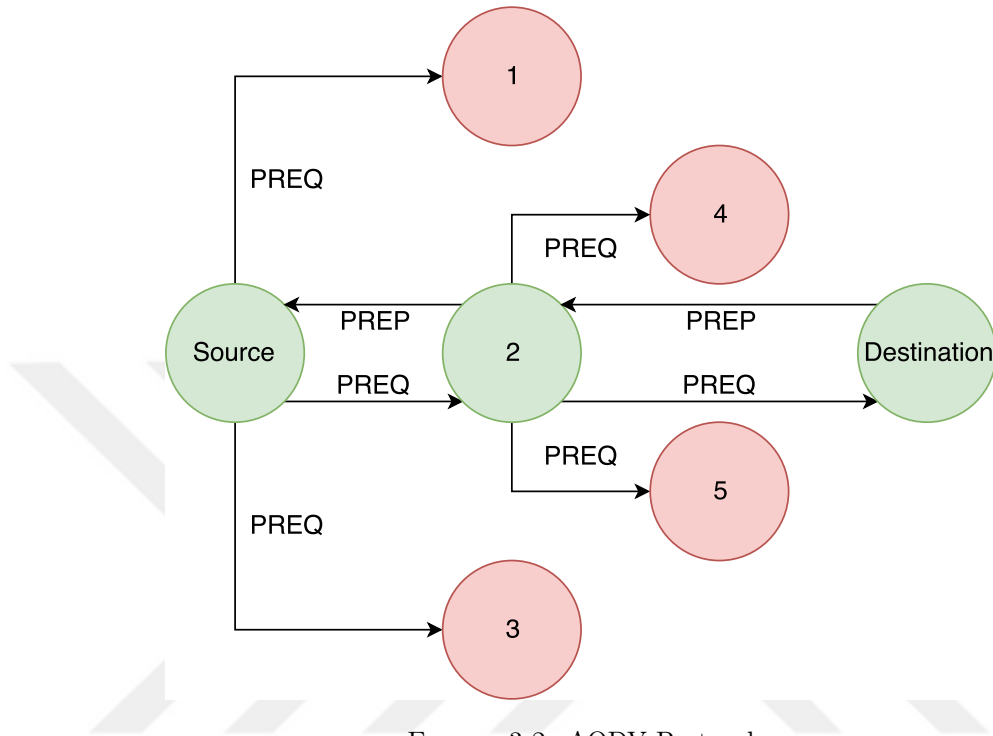


FIGURE 3.2: AODV Protocol

Wallgren *et al.* [22] implemented routing attacks by using Cooja simulator and Contiki operating system. Their preferred to use RPL protocol depending on the 6LoWPAN networks. Selective forwarding attacks, sinkhole attacks, HELLO flood attacks, wormhole attacks and sybil attacks are examined and implemented in the study. They also claimed RPL's weakness against routing attacks. Then they proposed a routing protocol to defend against selective forwarding attacks, named heartbeat protocol. They provided grounds to plan and implement IDS for the IoT.

Multihop, is another routing protocol, and is explained in [61]. It chooses the shortest way through the base station. Hop-count and power availability are the criteria [59]. Multihop is quite similar with MintRoute protocol. In the paper, two different methods of intrusion detection are explained before the presentation of their solution. Misused based approach, or signature based approach, give high accuracy results whereas it isn't good at detecting new attacks. An anomaly based approach is good at detecting new attacks but in many cases it gives high FAR. In their solution they propose, a hybrid

approach, where each node has a detection agent to identify suspicious nodes. If a suspicious node is detected by the detection agent, that is located on the node, an alert is sent to CA and the suspicious node is taken in blacklist temporarily. Then the last decision is the CA's. At the end of the paper, they present low RAM and ROM usage and high detection rates whereas they claim to use anomaly based approaches. But they didn't show and explain the implementation of the anomaly based approach.

The authors in [62] explain the vulnerable points of crowdsourced networks under a threat of independent attacks such as jamming or spoofing. They also present an attack detection system to secure civil air traffic control(ATC) using OpenSky. Additionally, they describe a trust model for connected sensors used in wireless air traffic networks and show insecurities in the design of ATC systems after briefly explaining new technology of ATC. Air traffic infrastructure is quite sensitive so injection of Ghost Aircraft, modification of labels, jamming or Denial of Service are some of wireless attacks to ATC. The presented system is based on an uncertified systems. ATC signals are verified validated to certified radar systems separately. Then with classical IDS system, available ATCs are collected and analysed. If there is a potential wireless attack, they are detected. OpenSky network is "community-based receiver network which continuously collects air traffic surveillance data" [63]. In April 2017, network has 260 registered and 300-450 anonymous sensors streaming data. Operators of registered sensors are known with personal contact but operators of anonymous sensors are unknown. Non-complex system for joining crowdsourcing network is so vital. Because, if the network expands, much more data can be collected. In the contrast, think about passport verification of identities could cause the owners to hesitate to supply data. Two kinds of threats are determined in the paper; outside threats who can make an injection, modifying or jamming signals and inside threats who can join crowdsourced networks and make ground to new attacks. For different attacks, different detection methods are examined. Mentioned detection methods are plausibility checks, cross referencing, multilateration and statistical analysis. After explaining the outside attacks, two insider attack are also examined. The first one is a single sensor data manipulation. The intruder targets to manipulate a signal sensor as the attack's name suggest. Second attack is Sybil Attacks where the intruder uses different, a normal, node's identity while communicating with neighbour nodes. Sybil node creates new ID of 2-bytes integers from previously stolen ID of a neighbour. It is more crucial then the first one [64]. Intruder targets to collect the multiple sensors under

his control then with these sensors, he feeds bogus data to OpenSky.

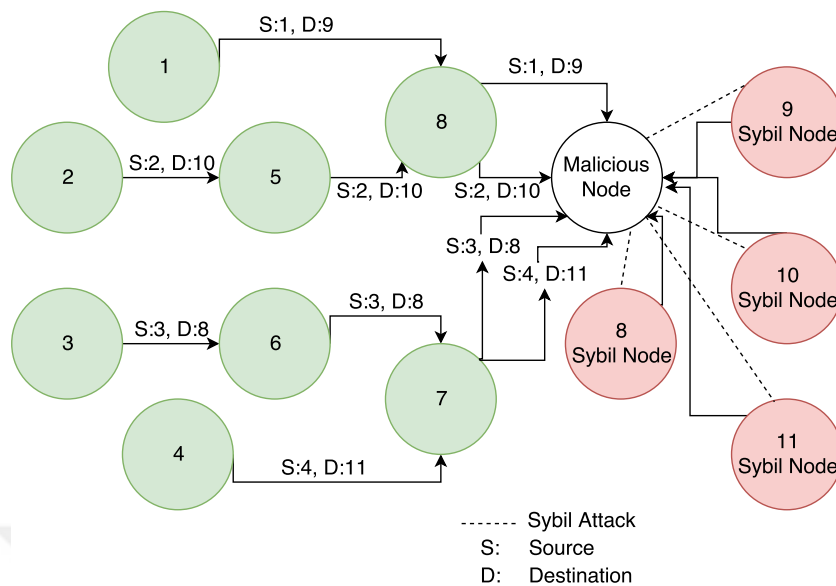


FIGURE 3.3: Sybil Attack

There are two approaches for predicting attacks to establish secure CPS; rule-based approach and behavioural approach.

In the rule-based approach, previous adversary attack signatures are used for identification of new adversary attacks. Even though it is faster than other approaches, it remains unresolved for the attacks that have different signature [65]. Wong *et al.* [66] proposed a rule-based solution, however false positive results are so high. Ilgun *et al.* [67] studied about state transition analysis on intrusion data. They set different signature for each state. If the system, named STAT, detects an activity near the sensitive state, alarm is risen.

New malicious activity can be predicted and a near real-time zero-day attack detection is made possible by the behavioural-based approach [68]. Duffield *et al.* [69] studied on ML based anomaly detection on IP flows. Mascaro *et al.* [12] tried to create an anomaly detection model for tracking vessels. The anomalies are identified by Bayesian networks. The study of Tsai *et al.* [19], includes a comparison of different ML techniques in the time between 2000 and 2007, a few of the techniques are naive Bayes networks, single classifiers, pattern classification, decision trees etc.

For secure IoT, detecting and if possible, predicting malicious attacks come into prominence to protect the IoT systems against attacks. There are many studies about routing

protocol attack detection for IoT with the rule-based approach. In the study of Raza *et al.* [70], node IDs and ranks are checked to matching assigned values for detecting anomalies. If a malicious node detected, an alarm is raised. The security of IoT nodes against the sinkhole attack is ensured by applying two rules that check the sender field of route update packets due to network topology [60]. However rule-based detection isn't efficient for complex systems and unidentified attacks because many rules are required which brings management difficulty of rules. Additionally, since rules are determined over known system and known attacks, new rules need to be added to deal with new kind of attacks.

Avram *et al.* [71] aim to detect routing attacks by using Self Organizing Maps (SOM) algorithm. For the AODV routing protocol, they created scenarios, benign and mixed, to calculate the proposed system accuracy. Their algorithm detects some routing attacks with high accuracy, nevertheless the study suffers from a high false positive rate (9.5%).

Banković *et al.* [26], researchers aim to detect unknown attacks in WSN. In the centralised routing tree, there are some PDA-like sensors, that have more power resource and computational capacity to solve node's constraints issue. They aim to detect Sybil attacks, a kind of routing attack that malicious nodes try to impersonate other nodes. Their proposed algorithm has been tested on a 40-node WSN, gets high detection rate until the percentage of malicious nodes exceed 52%.

Pongle *et al.* [24] studied on detecting wormhole attacks in IoT. Wormhole attack is a topology based routing attack that aims to affect the network topology and packet traffic flows and it is also depicted in Figure 3.4. Their proposed IDS-based detection system identifies the wormhole attacks by using node's and neighbour's location and also identifies attacker node by using received signal strength. They focus on RPL as a routing protocol and simulate the IoT network by using Cooja [2]. Their network topology has up to 24 nodes. The numbers of nodes aren't realistic for real IoT environment. Additionally, proposed rule-based system is inefficient against small mutations of know attacks. Likewise, Nait-Abdesselam *et al.* [25] proposed a rule-based detection system. They use OLSR [72], as routing protocol, and ns-2 simulator [73] for creating a network. In the study of Nait-Abdesselam *et al.* [25], the number of nodes are more than other studies [24], such as 10 to 50 nodes.

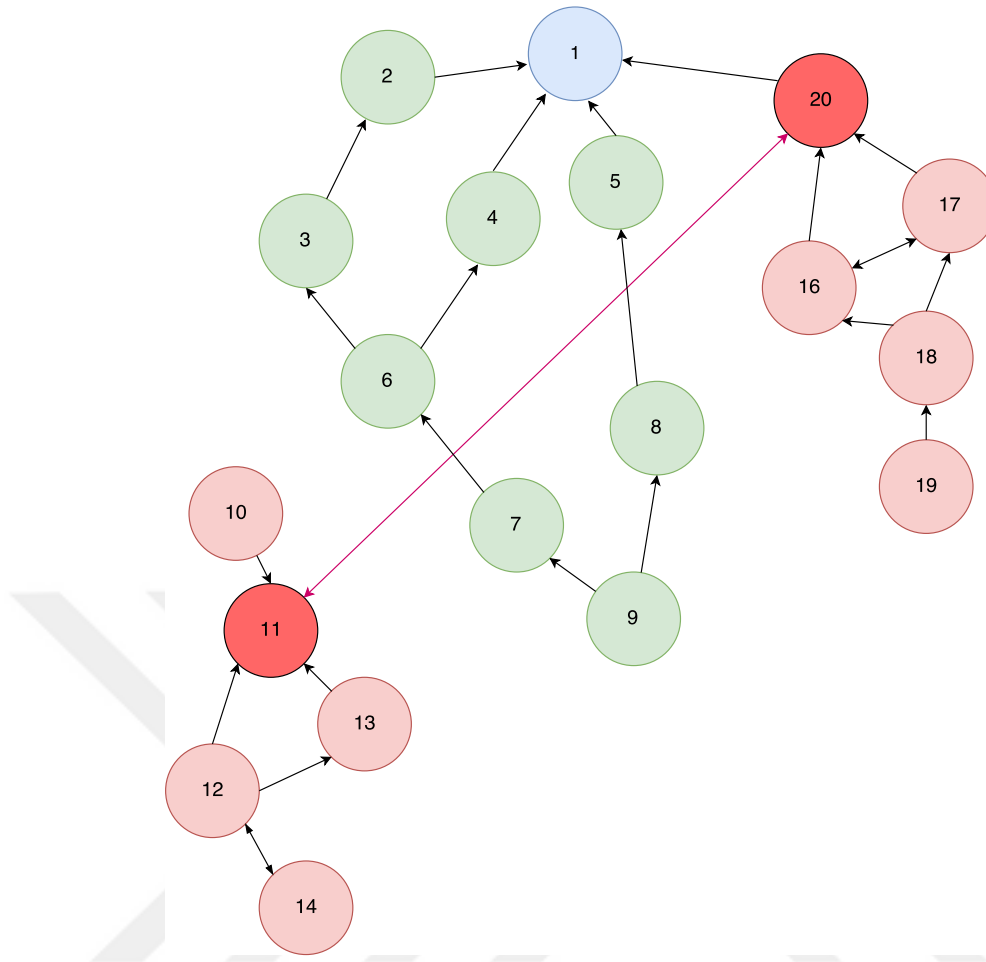


FIGURE 3.4: Wormhole Attack

Dhamodharan *et al.* [64] aim to detect sybil attack. In the sybil attack, briefly attacker uses different (normal) node's identity while communicating with neighbour nodes. They create a network, that is based on an AODV protocol, by using an ns-2 simulator [73]. Message authentication is applied to detect sybil node, as a rule-based solution.

The usage and efficiency of various ML and data mining methods for intrusion detection is discussed by Buczak *et al.* [17]. ML result metrics (False Positive, False Negative etc.), core ML methods (SVM, Bayesian, Decision Tree, Clustering etc.), complexity of the ML methods used on public datasets and features of the datasets are discussed.

As an example of SVM, in the research of Chowdhury *et al.* [20], ten subsets of dataset features are created randomly by feature selection, where one of the subsets contains three features. Then, these feature subsets are input to the support vector machine (SVM) algorithm respectively. The authors claim to improve detection accuracy from

88.03% to 98.76% as a result by using one of the feature subsets instead of using all features.

Classical ML methods have been applied to attack detection in recent studies. Branch *et al.* [74] applied k-NN algorithm to detect outliers in WSN. Janakiram *et al.* [11] also applied BBN algorithm to detect outliers in WSN but these studies have no contribution on routing attacks.

There are some applications of SVM to IoT. SVM is an supervised ML model for solving classification and regression problems. The researchers proposed outlier detection systems in WSN. However these studies don't address routing attacks, see refs. [14], [15], [16], [13].

In the study of Kaplantzis *et al.* [14], researchers aim to build a simple classification based IDS to detect selective-forwarding attacks. IDS generates an alarm depending upon bandwidth and hop count thresholds. The classification process is designed using SVM classifiers. Determined features, bandwidth and hop count, can produce accurate results for the blackhole attack, but for other routing attacks, they are not as efficient.

3.1 Deep Learning based Cyber Security Methods

Attack detection can be possible in two ways; rule-based and behavioral-based. Rule-based solutions give better results (high accuracy and low false alarm rate) for detecting attacks that already extent with the characteristics of the attack, additionally, this approach uses low energy in rule-based detection. Rule-based attacks focus the signature of the known attacks that enables the good results. However, rule-based attack detection remained weak against new attacks. Because, nearly 99% of cyber-attacks are the small mutations of the known cyber-attacks and just 1% of them are unique. So rule-based approaches couldn't distinguish the benign and attack behaviour against the mutation attacks. On the other hand, behavioral-based attack detection focus the behaviour of the attacks, if it detects any anomalous behaviour, as for that the normal behaviour of the system, it gives an alarm [75].

Classical ML methods have been applied to attack detection in recent studies. In [74], Branch *et al.* applied k-NN algorithm to detect outliers in WSN. Janakiram *et al.* [11] also

implemented BBN algorithm to detect outliers in WSN. Artificial Neural Networks(ANN) are also applied for intrusion detection [76], [77].

DL prove itself in the big data area that got significant results on classification and intrusion detection [78], [79] and [80]. Distributed attack detection is done by fog computing [37] in this study. They also used NSL-KDD [81] dataset for detecting attacks. A promising approach for distributed DL, but does not address IoT attacks specifically. Diro *et al.* [53], proposed a DL based distributed attack detection for IoT. They also compare the traditional ML with DL about performance on distributed attack detection and prove that DL is the state of the art for attack detection. Distributed attack detection is done by fog computing [37] in this study. They also used NSL-KDD [81] dataset for detecting attacks. A promising approach for distributed DL, but does not address IoT attacks specifically.

Chapter 4

IoT Routing Attack Dataset(IRAD)

First of all, we need the datasets that have routing attacks. In this research area, the lack of a dataset is one of the biggest challenges. So we simulated the routing attacks within different scenarios and processed raw datasets to make them ready to detection process. Subsequently, we transform the PCAP files to CSV with Wireshark. After that, a feature extraction process is applied to the generated CSV files by using our developed Python data preprocessing script. Finally we concatenate the same attack datasets to make a comprehensive dataset. Thus, we have generated an IoT Routing Attack Dataset(IRAD) for one of the purposes of our study. The details of mentioned scenarios and datasets are listed in Table 4.1.

TABLE 4.1: Details of Datasets

Datasets	Scenarios	No. of Nodes	<i>Malicious</i>		<i>Benign</i>		
			No. of M./N. Nodes	Total Packet Count	Scenarios	No. of Nodes	Total Packet Count
Decreased Rank	DR10	10	2/8	130.240	Benign10	10	121.047
	DR20	20	4/16	398.143	Benign20	20	120.897
	DR100	100	10/90	456.816	Benign100	100	150.078
	DR1000	1000	100/900	801.396	Benign1000	1000	739.845
Hello Flood	HF10	10	2/8	100.538	Benign10	10	121.047
	HF20	20	4/16	104.302	Benign20	20	120.897
	HF100	100	10/90	300.986	Benign100	100	150.078
	HF1000	1000	100/900	1.739.226	Benign1000	1000	739.845
Version Number	VN10	10	2/8	112.044	Benign10	10	121.047
	VN20	20	4/16	203.153	Benign20	20	120.897
	VN100	100	10/90	223.275	Benign100	100	150.078
	VN1000	1000	100/900	1.585.075	Benign1000	1000	739.845

In the simulation, if there is no malicious node, all nodes are normal, so the benign scenarios have the same values. We tried to give our best during dataset simulation process as in the whole thesis. Because the simulation process is getting harder when

the number of nodes increase in the network topology. For this reasons, the values of *Total Packet Count* is different from each other.

After the simulation (that explained in section 2.1.4.1), we need to apply some data engineering staff to the simulation. In this chapter, how we prepared the IRAD will be explained in detail.

4.1 Feature Extraction

The ML algorithms need some attributes about data for learning which are obtained by feature extraction.

After the scenarios are simulated, the datasets are produced as PCAP files. We dissected the PCAP file to CSV by using Wireshark and preprocessing section is fed by this CSV files. The data pre-processing step involves extracting useful features from the data for preventing over-fitting and to obtain problem oriented attributes.

We simulated different scenarios that have different network topologies and size for each type of attacks. In the result of the simulations we get the raw datasets. Before the feature extraction section, a sample of raw dataset is shown in Table 4.2.

TABLE 4.2: A Sample of Raw Dataset

No.	Time	Source	Destination	Length	Info
4755	14,611416	fe80::c30c:0:0:12	fe80::c30c:0:0:11	102	RPL Control (DODAG Information Object)
4756	14,611886	fe80::c30c:0:0:8	ff02::1a	97	RPL Control (DODAG Information Object)
4757	14,612891	fe80::c30c:0:0:4	fe80::c30c:0:0:18	76	RPL Control (Destination Advertisement Object)

Cooja exports PCAP and CSV files after the end of the simulation. However, the raw data files aren't sufficient to be the input to learning algorithm because the raw dataset includes information such as source/destination nodes address and packet length, which causes noise and overfitting in the learning algorithm. For this reasons we developed a feature extraction algorithm named Enrichment of IoT Raw Dataset, in Python by using Pandas[82] and Numpy [83] libraries. These libraries makes the mathematical operations, which are necessary for feature extracting, easier. We implemented a dictionary structure to deal with a large number of nodes. We opted not to calculate global statistics over total simulated time or total packet count since this kind of a calculation could decrease the importance of the main extracted features. So we have divided all the simulation to time frames, or windows of 1000 ms duration. Before this process, it is necessary to sort

the datasets by simulation time, because sequence of packet simulation time is highly important for feature extraction and Cooja extracts PCAP files in wrong time sequence. It happens especially for wide range network topologies and long simulation times. The pseudocode of our data preprocessing algorithm is also shown in Algorithm 1.

Algorithm 1 Enrichment of IoT Raw Dataset

function

array \leftarrow *RAWdataset.csv*

Sorted array

\triangleright Sorting by time

Feature conversion

Feature Extraction:

1000ms \leftarrow Windowing Size

Calculating Feature values within windowing size

Labelling the dataset

End of *the Feature Extraction*

End the function.

Raw datasets involve both quantitative and qualitative features. However, our learning algorithm accepts just quantitative values. So we applied feature conversion to qualitative features to transform their unified format.

We firstly convert the source and destination address from IPv6 format to Node id. For example:

$$fe80 :: c30c : 0 : 0 : 12 \implies 12 \quad (4.1)$$

The broadcast packets are handled as follows. In raw dataset, if the destination address is *ff02::1a*, that means the source node sends broadcast packets. This value is converted to *9999* to avoid any coincidence with another node:

$$ff02 :: 1a \implies 9999 \quad (4.2)$$

We also encoded the information of the packets as shown in Table 4.3. DAO is used in RPL for unicasting the destination information due to the selected parents. DIO is

the most important message type in RPL. It keeps the current rank of the node, and determines the best route through the base node by using specific metrics as distance or hop-count. Another message type is DIS. Nodes use DIS for joining to WSN. Ack is an acknowledgment message type for using to give responses by nodes [48]. These are encoded respectively: 1, 2 3, 4. Other types in our datasets are Protocol Data Unit (PDU) and UDP packets which are simulated data packets.

TABLE 4.3: Encode of Information Feature

Info	Encode Value
RPL Control (Destination Advertisement Object)	1
RPL Control (DODAG Information Solicitation)	2
RPL Control (DODAG Information Object)	3
Ack	4
Data Packets	5,6,7

First, we calculated the Transmitted and Received Packets Counts (TPC and RCP) for each node in 1000ms in a specified time frame. Then, we divide these values to 1000ms and get Transmission Rate and Reception Rate for each node, TR (4.2) and RR (2.1) respectively, for all time frames. Duration time for each packet transmission and reception are calculated. Total Transmission Time(TTT) and Total Reception Time(TRT) are calculated by adding up duration time of each transmission and reception packet in 1000ms. Then Transmission and Reception Average Time for each node, TAT (2.4) and RAT (2.2), are calculated. These features are listed in Table 4.4. Our last features are about control packets; DAO, DIO and DIS. Number of transmitted control packets of each node are calculated within the windowing size, 1000 ms.

TABLE 4.4: Extracted Features

Number	Name/Abbreviation	Description
1	No.	Packet sequence number
2	Time	Simulation time
3	Source	Source Node IP
4	Destination	Destination Node IP
5	Length	Packet Length
6	Info	Packet Information
7	TR	Transmission Rate
8	RR	Reception Rate
9	TAT	Transmission Average Time
10	RAT	Reception Average Time
11	TPC	Transmitted Packet Count
12	RPC	Received Packet Count
13	TTT	Total Transmission Time
14	TRT	Total Reception Time
15	DAO	DAO Packet Count
16	DIS	DIS Packet Count
17	DIO	DIO Packet Count
18	Label	Normal/Malicious Label

The labelling process is also important. In our datasets, attack packets are labelled 1 and benign packets are labelled 0. We labelled the datasets, which has malicious node and activity, 1 and labelled the benign datasets 0. Because, malicious nodes affect the entire network activity and influence normal node communication. For instance, most of the routing attacks may change the network topology. Accordingly, the routing path of normal nodes are changed.

A sample of the preprocessed dataset is shown in Table 4.5. This 10 packets are selected randomly from the Version Model 1000 dataset. Feature numbers are increased respectively from 1 to 18.

After feature extraction process, the datasets include up to 26×10^6 data items which shows why a deep learning based methodology is needed.

TABLE 4.5: Sample Dataset

No.	Time	Src	Dst	Lngh	Info	TR	RR	TAT	RAT	TPC	RPC	TTT	TRT	DAO	DIS	DIO	Label
620.851	48,455665	94	893	76	1	0,197	0,197	197	197	0,012	0,012	0,00006	0,00005	197	0	0	1
620.852	48,455694	389	339	76	1	0,096	0,096	96	96	0,005	0,005	0,00005	0,00005	96	0	0	1
620.854	48,455762	158	620	76	1	0,196	0,166	196	166	0,01	0,008	0,00005	0,00005	166	0	30	1
620.855	48,455774	971	271	76	1	0,22	0,22	220	220	0,008	0,008	0,00004	0,00004	220	0	0	1
620.856	48,455779	994	331	76	1	0,227	0,227	227	227	0,008	0,009	0,00004	0,00004	227	0	0	1
620.857	48,455782	354	894	76	1	0,284	0,128	284	128	0,006	0,006	0,00005	0,00005	284	0	0	1
620.867	48,455816	565	9999	97	3	0,03	1,7	30	1698	0,002	0,097	0,00007	0,00006	0	0	30	1
620.858	48,455836	808	792	76	1	0,263	0,189	263	189	0,01	0,006	0,00004	0,00003	189	0	74	1
620.861	48,455991	691	134	76	1	0,19	0,19	190	190	0,012	0,012	0,00006	0,00006	190	0	0	1
620.874	48,456005	430	33	102	3	0,171	0,171	171	171	0,013	0,013	0,00006	0,00006	0	0	171	1

Briefly, we simulated scenarios as shown in Table 4.1. Then we extracted features. Finally, we mixed the malicious and benign dataset which has the same network topology. For example, Version Number 100 and Benign 100 datasets are mixed.

Additionally, Foren6 [84] is a 6LoWPAN network and protocol(RPL, IPv6,...) analysis tool. It uses the captured file by the sniffers like a The PCAP file from wireshark. Network topology is also visualized and Foren6 also gives statistics of each node in the network as total sent control packets, valid life time, MAC address, rank value in the network etc. Foren6 user interface is shown in Figure 4.1. We also use this analyses tool to check our extracted feature values such as DIO, DAO, DIS.

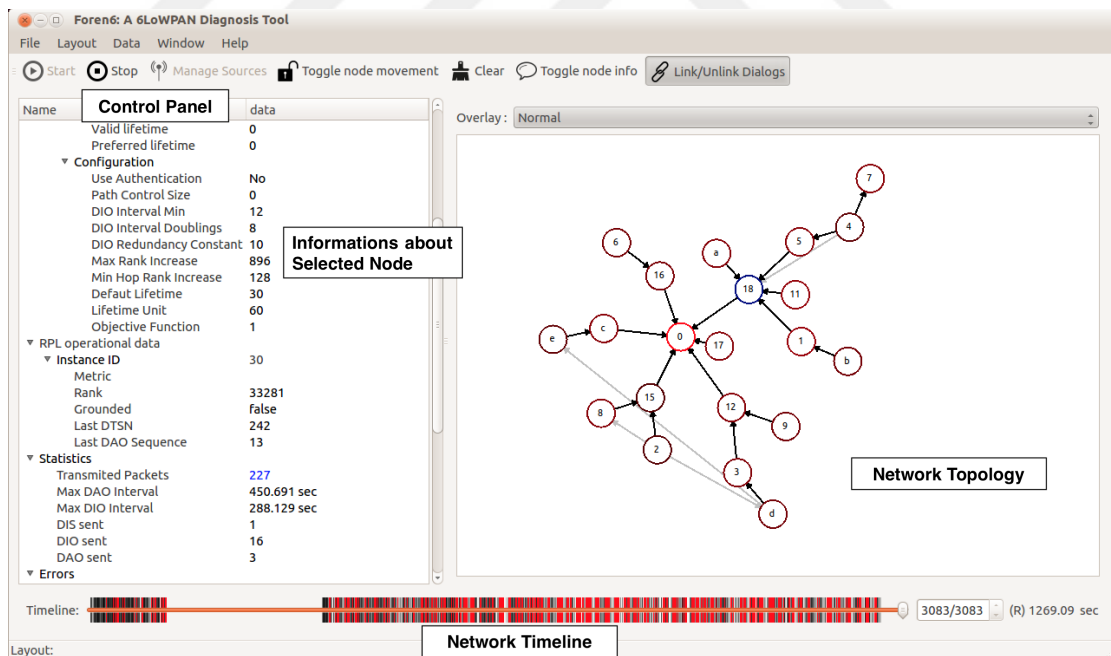


FIGURE 4.1: Foren6 User Interface

4.2 Feature Normalization

Feature normalization is a pre-processing method for scaling all values of each feature into a certain range. It makes the data smoother and cleans the bias from data, ensuring high accuracy rate [85].

We get different datasets from different scenarios for each routing attack. The datasets have different data values in different ranges due to their network topology. In this situation, datasets don't give relevant results to us and the learning algorithm couldn't work effectively. So we performed feature normalization for pulling all datasets in the same range by using our data normalization algorithm. We applied quantile transform and min-max scaling to datasets, respectively [86]. Each feature is enforced the normal quantile transform, separately. The transform aims to spread marginal values, it may change correlation between the values. However difference between the values, which are directly comparable and useful for the learning algorithm, are more significant. Then we scale all values in the datasets to range 0-1 by min-max scaling. The effect of feature normalization process to features are depicted in 4.2 and 4.3. The figures are transmission rate of Decreased Rank 20.

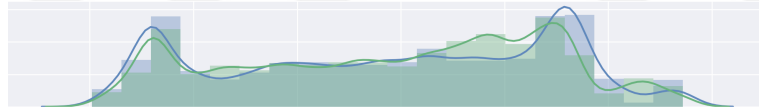


FIGURE 4.2: Transmission Rate before Feature Normalization Process

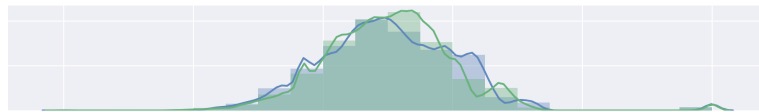


FIGURE 4.3: Transmission Rate after Feature Normalization Process

Finally we concatenated all datasets, that have different network topology, for each attack. So we get three IoT datasets, as clearly seen in Table 4.1. The pseudocode of our data normalization algorithm is also shown in Algorithm 2.

Algorithm 2 Data Normalization Algorithm**function**Mixed Dataset \leftarrow Benign, Malicious Dataset*Feature Normalization:*Quantile Transform Function \leftarrow Mixed Dataset*Min Max Scale Function* \leftarrow *Transformed Dataset*End of *the Feature Normalization*IoT Dataset \leftarrow Mixed Datasets \triangleright concatenating the datasets*End the function.*

4.3 Feature Importance and Selection

Feature selection is a key step in ML. Feature selection is generally applied to the dataset before running the ML algorithm, because it eliminates the irrelevant, weakly relevant features and selects the optimal subset of all features. It identifies the proper subset of all data features and makes the data serviceable. There are two main challenges; the large size of data and its inconvenient form. A dataset has two dimensions; number of instances and number of features which are usually way too large. This huge volume also brings a complexity. On the other side, datasets are created without the features or attributes. Particularly, the network datasets are captured from the Internet or closed networks as PCAP form.

We used a combination of random decision trees, histograms and pearson coefficient correlation [28] for feature selection process.

We evaluated the importance of the extracted features by using a number of randomized decision trees (extra trees). The main idea of randomized decision trees is bagging the means to average noisy and unbiased models to create a model with a lower variance. Random decision trees work as a large collection of uncorrelated decision trees. In brief, randomized decision trees create different decision trees and extract importance of features by comparing the created trees [27]. Subsequently we determine the efficient features. If the importance is high, that means the node dilutes the effect of other nodes and it may cause over-fitting at the learning process. By the way, different feature selection process' should apply for each problem. Because, feature importance rates are

TABLE 4.6: Feature Importance of Decreased Rank Attack

Name/Abbreviation	Feature Importance Rate	Selected Feature
No.	0.018604	No
Time	0.018300	No
Source	0.838269	No
Destination	0.011982	No
Length	0.003977	No
Info	0.004729	No
TR	0.011376	Yes
RR	0.008440	Yes
TAT	0.008773	Yes
RAT	0.006830	Yes
TPC	0.023076	Yes
RPC	0.007300	Yes
TTT	0.012467	Yes
TRT	0.008323	Yes
DAO	0.009123	Yes
DIS	0.000328	No
DIO	0.010139	Yes

different due to dataset content. Decreased rank attack importance rates are listed in Table 4.6 as a sample. As clearly seen, some features dilute the other features. The rates in the tables are first importance rates. When the most significant feature is dropped, the importance rates change.

In the feature selection process, we dropped the features with the most and least importance. We evaluated the importance of selected features again. Feature importance rates for decreased rank attack is shown in Figure 4.4, as a sample. X axis represents abbreviation of selected features. Y axis represents the importance rate of features. We dropped the first five features and keep DIS features while creating hello-flood attack detection model, because of the dropped features are close to each other and are very high(first five).Than we keep the DIS feature, because nodes use DIS packets to send broadcast messages in RPL. For version number attack, we applied the same process with Decreased Rank model. Briefly, we dropped mentioned features that have highest and lowest importance rates. Because when we tried to keep them while model training, they caused overfitting.

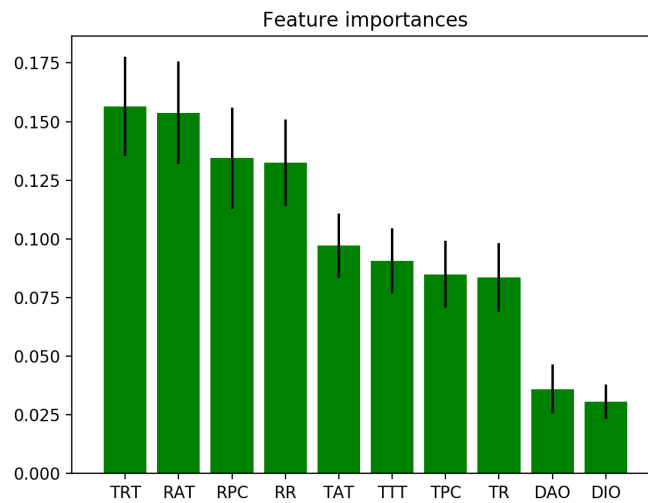


FIGURE 4.4: Feature Importance for Decreased Rank Attack after Feature Selection

We also extracted the dataset histograms to observe differences of 0's and 1's within each feature. If the line of 0 label doesn't follow the line of 1 label in a feature, it is a significant feature for learning algorithm. Otherwise, the feature is insignificant, the learning algorithm couldn't use the feature, effectively. A sample histogram of significant and insignificant features from Version Number Attack dataset is showed in Figure 4.5 and 4.6, respectively. The significant sample belongs to Transmission Rate and the insignificant sample belongs to Packet Length.

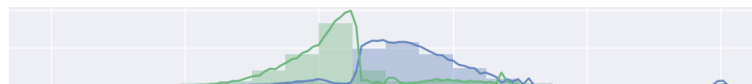


FIGURE 4.5: Significant Feature Histogram of Version Number Attack Dataset

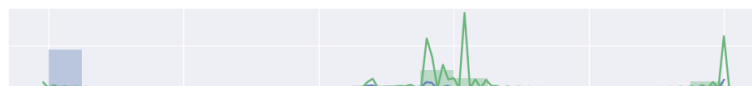


FIGURE 4.6: Insignificant Feature Histogram of Version Number Attack Dataset

We also evaluated the pearson rate of our datasets to measure the correlation between features. The pearson coefficient correlation is used for understanding how strong dependency is there between features and data. Guyon *et al.* [28] said that the pearson coefficient correlation is appropriate for binary classification problems. Additionally, using pearson coefficient correlation gives some information about linearity and nonlinearity of our datasets to us. Pearson coefficient correlation has a value within 1 and

-1. 1 means total positive linear correlation, -1 means total negative linear correlation and 0 means nonlinear correlation. Basic equations of pearson coefficient correlation is shown in 4.3. In the equation, cov is the covariance, are the standard deviation of x and y , respectively. Standard deviation of x and y are in the denominator, respectively. It is more popular than a basic equation. Additionally, pearson rate of total transmission time feature is also showed in Figure 4.7

$$\rho_{x,y} = \frac{cov(x,y)}{\sigma_x * \sigma_y} \quad (4.3)$$

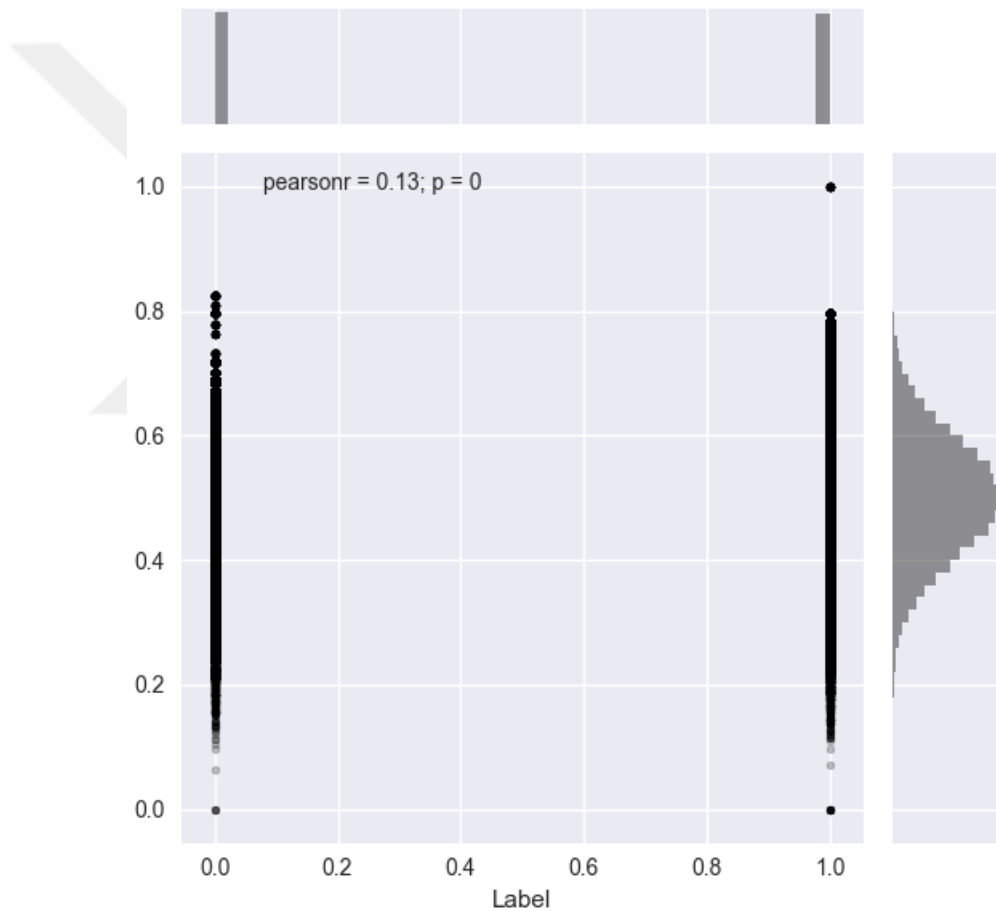


FIGURE 4.7: Pearson Rate of Total Transmission Time Feature

4.4 Overview of Datasets

We created three datasets that are Decreased Rank Attack Dataset, Hello-Flood Attack Dataset and Version Number Attack Dataset. They consist both of attack and benign

activities. The number of values what they include and the size of attack dataset files are listed in Table 4.7

TABLE 4.7: Datasets with Numbers

	Number of Values	Size (Gb)
Decreased Rank Attack Dataset	49,873,385	0.58
Hello-Flood Attack Dataset	64,179,435	0.75
Version Number Attack Dataset	22,868,210	0.27

After the dataset generation, we created an IoT dataset and named it IRAD(IoT Routing Attack Dataset). We also compared IRAD with most popular public datasets that are preferred to be used in cyber security researches. Results of the comparison are listed in Table 4.8.

TABLE 4.8: A Comparison of Datasets

Parameters	UNSW-NB15	KDDCUP99	IRAD
Simulation	Yes	Yes	Yes
Attack Types	9	4	3
Number of Networks	3	2	16
Data Type	Peap files	3 types (tcpdump, BSM and dump files)	Pcap files
Feature Extraction	Argus, Bro-IDS and new tools	Bro-IDS tool	Own Feature Extraction Algorithm
Extracted Features	49	42	18
No. of distinct ip address	45	11	4520

The comparison is the proof that IRAD is the most scalable dataset, considering *No. of distinct ip address*. We also supported our deep learning process with the products of 16 different networks, considering *Number of Networks*. Finally, IRAD is a public IoT attack dataset, that makes it novel.

Chapter 5

Deep Learning Based Detection of Routing Attacks

In this section, our deep learning based routing attack detection will be explained. We evaluate the importance of the features due to the datasets' index for selecting features to make the learning process more accurate. The features with too high and low importances are dropped in order to prevent overfitting. The datasets are normalized by a feature normalization process to make the training process faster. The output of the feature preprocessing steps are preprocessed datasets which are taken into the deep learning algorithm. The learning algorithm is implemented by the help of Python libraries such as Keras [87], Scikit [86] and Numpy [83]. The learning process outputs the IoT attack detection model. We tested the model against multiple test scenarios for more accurate measurement of precision and recall.

5.1 Routing Attack Detection

Routing attacks can be detected by signature based solutions and anomaly solutions. Signature based solutions are better against routing attacks that have little change of its nature. So anomaly based solutions are better than signature based solutions about the detection accuracy of new attacks. Diro *et al.* [53] explained in their research that deep learning has better performance than shallow learning. In the light of this brief information, we used deep learning to detect routing attacks.

5.2 Deep Learning Model

In this study, Keras [87] is utilized as DL framework since it ensures many advantages like its modularity, and since it enables us to build and test a complex neural networks quickly. First, Keras is an open source DL library that integrated into the Python ecosystem. Second, Keras also has wide range community and a detailed user manual. After that, Keras enables to build complex neural network easily.

Tensorflow [88] is a framework that is developed by Google for DL. It is suitable for working with CPUs and GPUs. Tensorflow also contains several implementations to build complex DL models. It is runnable on Mac, Windows and linux, as a python package. It is also utilities with Keras.

Firstly, we shuffled the dataset to improve the deep model performance and avoid overfitting. Before building the neural layers, the preprocessed dataset are split into two parts. First one, X, is the part without label feature and the other one is, Y. Briefly, Y is the part that make this learning algorithm to supervised learning. X and Y are split two part; X_{train} , X_{test} , Y_{train} and Y_{test} . Train parts are used in training section. Test parts are used to evaluate the performance of training process.

For creating a deep neural model, we used a sequential model, Ada Delta [89] function [90] as an optimizer and mean squared error (MSE) as a loss function. After the training process, we save the created model and weights as a json file. Finally, we tested our model for routing attacks detection by applying model.predict function on other datasets. The pseudocode of our DL algorithm is also shown in Algorithm 3.

Algorithm 3 Deep Learning Algorithm

```

1: function
2:   dftrain ← dataset.csv
3:   Shuffling dftrain row by row
4:   X, Y ← dftrain                                ▷ Splitting dataset for learning
5:   Feature Importance and Selection:
6:   Fit (X, Y) in Randomised Decision Trees
7:   Features are dropped                            ▷ in consideration of Randomised Decision Trees and
   Histograms
8:   End of the Feature Importance and Selection
9:   Define X train, Y train and X test, Y test
10: Deep Learning Model:
11:   model ← Sequential Model
12:   Set Neural Network Layers
13:   optimizer ← Ada Delta Optimizer                ▷ Set Optimizer
14:   Compile Classifier as MSE
15:   model ← X train                                ▷ Training starts
16:   Save Model and Weights as json file
17: Prediction:
18:   Load the datasets as dfpredict
19:   Xpredict, Ypredict ← dfpredict
20:   X predict ← Scaled X predict
21:   Classification Report exports
22: End the function.

```

We build our Neural Network as 7 layer. First layer is Input Layer that has 10 neuron. Number of input layer should be equal to the number of features (columns) in the dataset. The last layer is the output layer that has just 1 neuron. This is called a Regression Model. Our neural network has 5 hidden layers. First and fifth layers have 50 neuron. Second and fourth layers have 100 neuron. Third layer has 300 neuron. The neural network layers are depicted in Figure 5.1. Before the training starts, dataset is split again at the rate of 0.3 as validation dataset to tune the training performance of the

model. Our neural network model is shown in Figure 5.1. The number of neurons in hidden layers are like a triangle for reason is explained in Section 2.2.2.

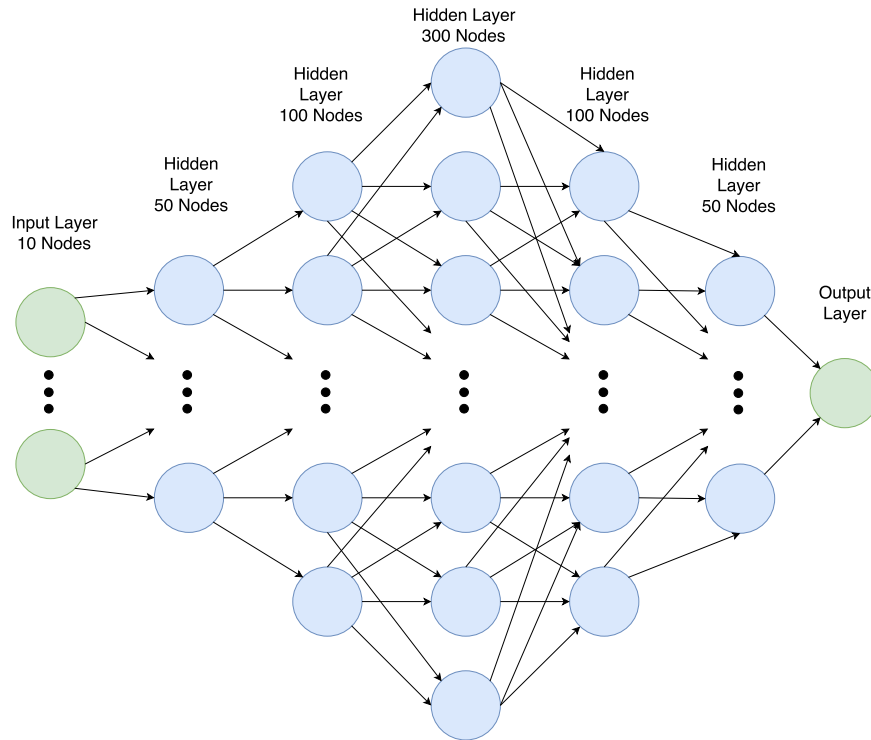


FIGURE 5.1: Deep Neural Network Layers

We use sigmoid function as activation function for output layer. Other functions (we get 0% accuracy rate) are not suitable for our problem and dataset except for softmax and tanh function. We get 35.3% and 35.8% accuracy rate when we use softmax and tanh function in output layer, respectively. Sigmoid function has a characteristic 'S' curve and mathematical representation is shown at 5.1. The accuracy of model is 98% when we use sigmoid function in output layer. In training process, we also avoid sharp increases from approximately 62% to 98% by applying dropout and regularization. We set dropout rate to 10% to consume less energy. Because, in every epoch, to dropout randomly selected nodes brings extra cost and time.

$$y = \frac{1}{1 + e^{-x}} \quad (5.1)$$

Additionally we also apply bias regularization to diminish nonlinearity of training by limiting the peak results. So the coefficient of our deep layers are much smaller. In brief,

connections between neurons generally cause overfitting when dealing with very large datasets. For this reasons we apply dropout and regularization.



Chapter 6

Evaluation and Conclusion

In this chapter, the performance evaluation and analysis of the deep learning based routing attack detection model will be explained briefly and contribution of the study, some recommendation and ideas of future work will be given.

6.1 Performance Evaluation

In attack detection, as well as many topics, obtaining high prediction accuracy rate(6.1) and low error rate(6.2) are main goals. The system's prediction is true, result is True, otherwise it is called False. If the prediction is about being attack, this situation is called Positive, otherwise is Negative. So there is four possibilities; prediction is true and attack, true and benign, false and attack, false and benign, True Positive (TP), True Negative(TN), False Positive(FP) and False Negative(FN), respectively. We preferred the accuracy and error (or loss) rate for evaluating of our deep learning based attack detection models' training performance.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (6.1)$$

$$Error = 1 - Accuracy \quad (6.2)$$

The precision (6.3) tells that how many of the attacks are detected by model, recall (6.4) tells that how many of the attack detection are correct and F1-Score (6.5) is, basically, harmonically average of both, precision and precision. We preferred these performance metrics to evaluate our deep learning based attack detection models' testing performance, means the performance against the different datasets. Performance metrics of normal activity aren't preferred to show, attack detection metrics usually are presented in many studies. We preferred to show normal activity performance of our models, too.

$$Precision = \frac{TP}{TP + FP} \quad (6.3)$$

$$Recall = \frac{TP}{TP + FN} \quad (6.4)$$

$$F1 - Score = \frac{2 * TP}{2 * TP + FP + FN} \quad (6.5)$$

6.2 Analysis

We created IRAD(IoT Routing Attack Dataset) that will fill the biggest gap in this research area. We compare our datasets with other novel and popular datasets, UNSW-NB15 [91] and KDDCUP99 [81] that listed in Table 4.8 in Chapter 4.4.

Briefly, we produced three IoT dataset that include 3 different routing attacks. Than we trained the neural networks with the preprocessed datasets of each routing attack. So, the weights of neural networks take different values due to the problems. After that, we got 3 IoT attack detection models for each routing attack. The models are also generalizable by taking into account different network topologies. The training performance of these models (Training Accuracy, Training Loss) are listed in Table 6.1. Training accuracy and loss of Hello Flood Model are better than other models. Because, the features, which we extracted, are matched successfully to the hello flood attack problem. On the other hand, detecting hello flood attacks with the features is easier than other attacks.

We tested our deep learning based attack detection models with the multiple datasets to sure about our models fidelity. Multiple datasets means that they include different

TABLE 6.1: Training Performance on Original Dataset

Model Name	Training Accuracy	Training Loss
Decreased Rank Model	94.9%	5%
Hello Flood Model	99.5%	0.8%
Version Number Model	95.2%	4.1%

datasets which have different number of nodes and different design of topology. We prefer to test with multiple datasets, because, that would prove our attack detection models are scalable.

Training accuracy and loss figures of all models are also given in the following subsections. Each figures includes two, different colored, curves. Blue one represents training accuracy whereas other, orange, represents validation accuracy. The results belongs to different datasets; training set and validation set. Test curve is more important than other for understanding the training performance. Because, test curve is the result of tuning process by using validation set of training dataset, as mentioned in previous chapter. In the figures test curve are better than training curve. This is also the proof of that overfitting was avoided. Because if training result better than validation results, it is the sign of overfitting.

6.2.1 Decreased Rank Attack

We tested our deep learning based Decreased Rank Attack Detection Model with the multiple datasets and performance metrics are listed in Table 6.2.

TABLE 6.2: Performance of Decreased Rank Model

	Precision	Recall	F1 Score
0	0.94	0.93	0.93
1	0.96	0.98	0.95
Average	0.95	0.96	0.94

Figure 6.1 and 6.2 are model training graphs of Decreased Rank Model.

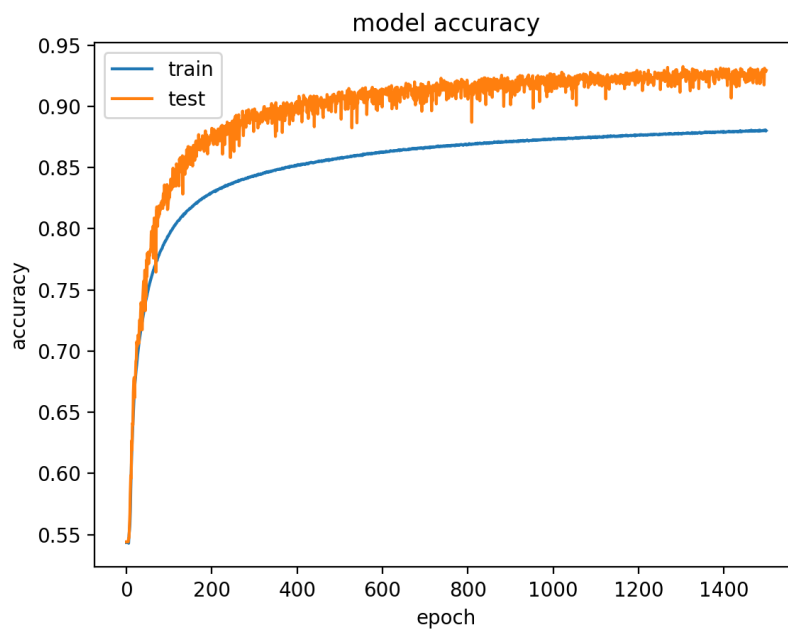


FIGURE 6.1: Model Accuracy of Decreased Rank Dataset

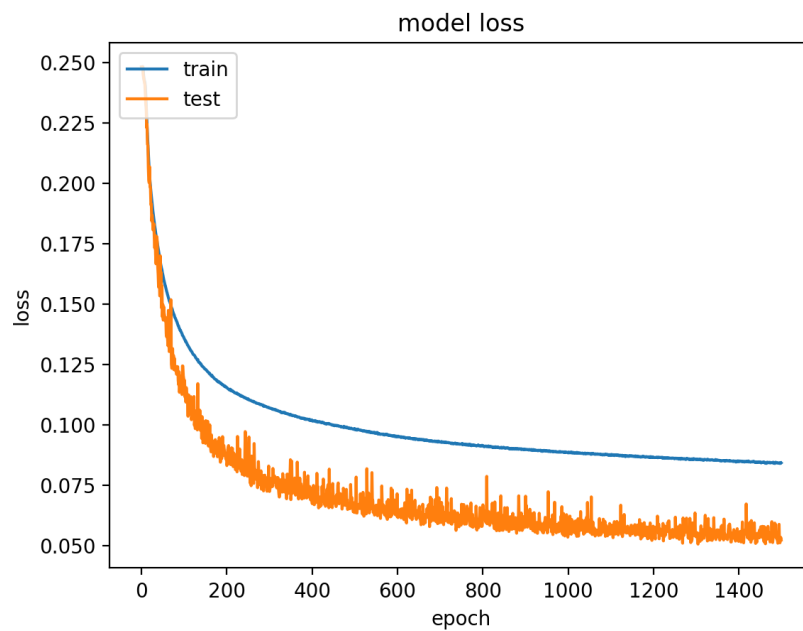


FIGURE 6.2: Model Loss of Decreased Rank Dataset

6.2.2 Hello Flood Attack

We tested our deep learning based Hello Flood Attack Detection Model with the multiple datasets and performance metrics are listed in Table 6.3.

TABLE 6.3: Performance of Hello-Flood Model

	Precision	Recall	F1 Score
0	0.97	0.98	0.99
1	0.98	0.96	0.98
Average	0.98	0.97	0.98

Figure 6.3 and 6.4 are model training graphs of Hello-Flood Model.

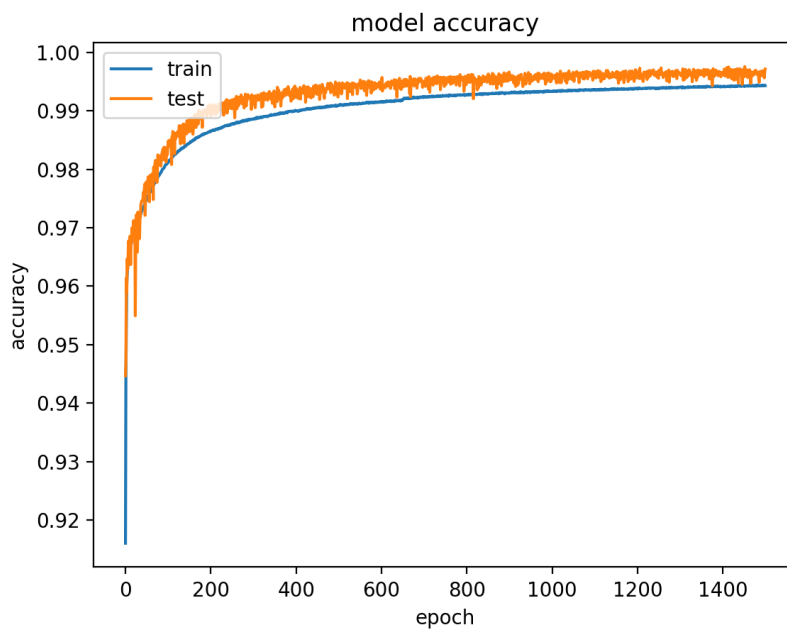


FIGURE 6.3: Model Accuracy of Hello-Flood Dataset

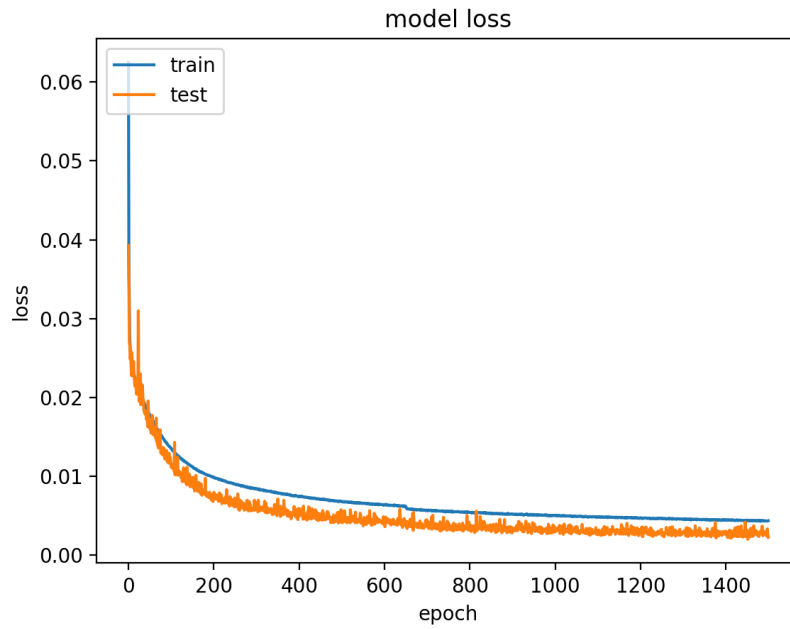


FIGURE 6.4: Model Loss of Hello-Flood Dataset

6.2.3 Version Number Attack

We tested our deep learning based Version Number Attack Detection Model with the multiple datasets and performance metrics are listed in Table 6.4.

TABLE 6.4: Performance of Version Number Model

	Precision	Recall	F1 Score
0	0.94	0.93	0.93
1	0.95	0.95	0.95
Average	0.94	0.94	0.95

Figure 6.5 and 6.6 are model training graphs of Version Number Model.

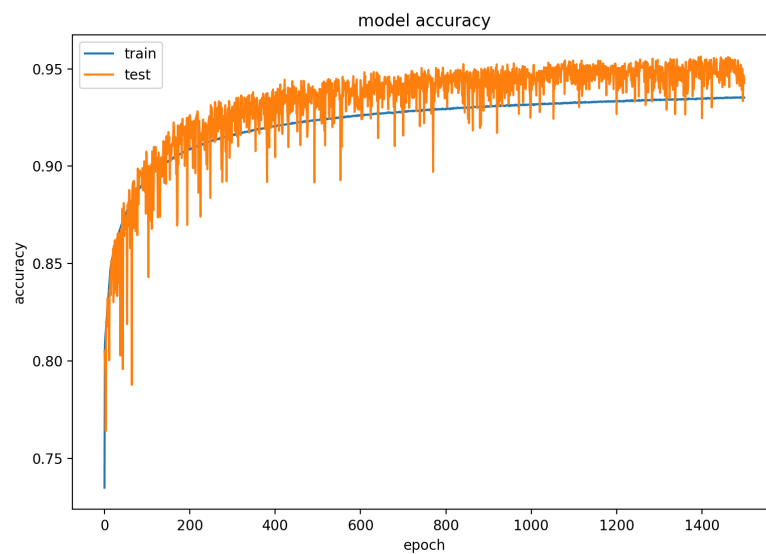


FIGURE 6.5: Model Accuracy of Version Number Dataset

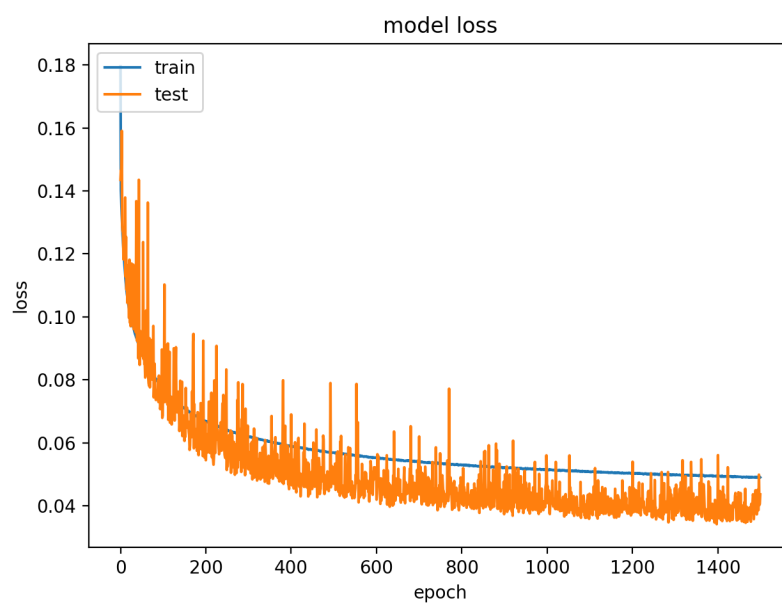


FIGURE 6.6: Model Loss of Version Number Dataset

Training epochs are sufficient as clearly seen in the figures. Because towards the end of epochs, increments in the accuracy figures and decrements in the loss figures are minor, little if any. In the end, we had 3 different model for each routing attack. We applied different feature selection process to each of the problem datasets. Then we trained the neural networks with the preprocessed datasets. After that, we got 3 IoT attack

TABLE 6.5: Performance of the Models over Multiple Datasets

Model Name	F1-Score
Decreased Rank Model	94.7%
Hello Flood Model	99%
Version Number Model	95%

detection models. The performance of these models are listed in Table 6.5 as F1-Score. Model's performance values are a proof of our models' scalability and accuracy.

6.3 Conclusion

This thesis is the proof of concept that deep learning can successfully deal with IoT security. The routing attacks (decreased rank attack, hello-flood attack and version number attack) are easily detected by our proposed attack detection models. This thesis also fills a highly important gap of the routing attack detection for IoT. The biggest issue of this kind of areas is the lack of datasets and also the data is not preferable when it has unrealistic content. Our attack datasets simulated by using real-codes of simulation tools. The datasets, which we produced and also preprocessed. Actually, the biggest effort of this work was generating and processing the attack datasets. In this thesis, we made this effort. The datasets include up to 64.2 million values. Additionally, we constructed a deep neural network, trained them with the produced routing attack datasets and create three attack detection models. Performances of the models are approximately up to 99%.

6.4 Future Works

The dataset has three routing attacks; decreased rank attack, hello flood attack and version number attacks. We are planning to enrich our IoT attack dataset by adding new routing attacks. We aim to increase the model prediction performance of three routing attacks and include more routing attacks in the study. We are also planing to create one deep neural network model to detect multiple attacks. By the way, we will diversify the scenarios with scenarios that have different rate of malicious and normal nodes.

Appendix A

```
#####  
# Script 1 - Enrichment of IoT Raw Dataset #  
#####  
  
# import the required packages  
import division  
...  
  
#readingsourcefile  
filename = 'VN1000%10RAW.csv'  
df = pd.read_csv(filename, sep=',', header=None, error_bad_lines=False)  
# removing first row of data then sorting according to time field  
array = df.as_matrix()  
arr = np.delete(array, 0, axis=0)  
array = sorted(arr, key=lambda x: float(x[1]))  
  
# storing each packet's transmission duration  
packetDurations = []  
  
# naming values of info field  
infoList = ['RPL Control (Destination Advertisement Object)',  
            'RPL Control (DODAG Information Solicitation)',  
            'RPL Control (DODAG Information Object)']
```



```
# Calculating node counts in per seconds #

packetsInEachSecond =
currentSecond = 0.0
lastSecond = floor(float(array[-1][1]))
while currentSecond <= lastSecond:
    currentSecondArray = []
    currentRow = 1
    while currentRow < len(array):
        currentPacketSecond = floor(float(array[currentRow][1]))
        if currentPacketSecond == currentSecond:
            currentSecondArray.append(array[currentRow])
        currentRow += 1
    packetsInEachSecond[currentSecond] = currentSecondArray
    currentSecond += 1.0

# Calculating control packet counts in per seconds #

sourceCountsBySeconds =
destinationCountsBySeconds =
daoCountsBySeconds =
disCountsBySeconds =
dioCountsBySeconds =
totalDaoDisDioBySecond =
for sec in packetsInEachSecond:
    sourceNodeCounts =
    destinationNodeCounts =
    daoCountsByNode =
    disCountsByNode =
    dioCountsByNode =
    for packet in packetsInEachSecond[sec]:
        src = packet[2]
```

```

dst = packet[3]
info = packet[5]
sourceNodeCounts[src] = 1 if src not in sourceNodeCounts else
sourceNodeCounts[src] + 1
destinationNodeCounts[dst] = 1 if dst not in destinationNodeCounts
else destinationNodeCounts[dst] + 1
if info == 1:
    daoCountsByNode[src] = 1 if src not in daoCountsByNode
    else daoCountsByNode[src] + 1
elif info == 2:
    disCountsByNode[src] = 1 if src not in disCountsByNode
else disCountsByNode[src] + 1
elif info == 3:
    dioCountsByNode[src] = 1 if src not in dioCountsByNode
else dioCountsByNode[src] + 1
if info == 1 or 2 or 3:
    totalDaoDisDioBySecond[sec] = 1 if sec not in totalDaoDis-
DioBySecond else totalDaoDisDioBySecond[sec] + 1
sourceCountsBySeconds[sec] = sourceNodeCounts
destinationCountsBySeconds[sec] = destinationNodeCounts
daoCountsBySeconds[sec] = daoCountsByNode
disCountsBySeconds[sec] = disCountsByNode
dioCountsBySeconds[sec] = dioCountsByNode

# Calculating total duration times by seconds #

transTotalDurBySec =
for sec in sourceCountsBySeconds:
    transTotalDurBySec[sec] =
    for node in sourceCountsBySeconds[sec]:
        transTotalDurBySec[sec][node] = 0
rcvTotalDurBySec =
for sec in destinationCountsBySeconds:

```

```

rcvTotalDurBySec[sec] =
for node in destinationCountsBySeconds[sec]:
    rcvTotalDurBySec[sec][node] = 0
counter = 1
while counter < len(packetDurations):
    second = packetDurations[counter][0]
    try:
        nodeTr = packetDurations[counter][1]
        nodeRcv = packetDurations[counter][2]
        transTotalDurBySec[second][nodeTr] = transTotalDurBySec[second][nodeTr]
+ packetDurations[counter][3]
        rcvTotalDurBySec[second][nodeRcv] = rcvTotalDurBySec[second][nodeRcv]
+ packetDurations[counter][3]
    except:
        print("This is an error message!")
    counter += 1

# Creating enriched IoT dataset file #

monitorArray = []
counter = 0
labelCounter = 0
while counter < len(array):
    if not pd.isnull(array[counter][2]):
        src = array[counter][2]
        dst = array[counter][3]
        row = array[counter]
        second = floor(float(array[counter][1]))
        srcCount = 0.0
        dstCount = 0.0
        if second in sourceCountsBySeconds:
            srcCounts = sourceCountsBySeconds[second]
            if src in srcCounts:

```

```

srcCount = srcCounts[src]
if second in destinationCountsBySeconds:
    dstCounts = destinationCountsBySeconds[second]
    if dst in dstCounts:
        dstCount = dstCounts[dst]
transmissionRate = srcCount / 1000.0
receptionRate = dstCount / 1000.0
TrRr = transmissionRate / receptionRate if receptionRate != 0 else
0

trnTtlDur = transTotalDurBySec[second][src]
rcvTtlDur = rcvTotalDurBySec[second][dst]
trnAverageTime = trnTtlDur / srcCount
rcvAverageTime = rcvTtlDur / dstCount

daoSrcCount = 0
disSrcCount = 0
dioSrcCount = 0
if len(daoCountsBySeconds[second]) > 0 and src in daoCountsBySeconds[second]:
    daoSrcCount = daoCountsBySeconds[second][src]
else:
    daoSrcCount = 0
if len(disCountsBySeconds[second]) > 0 and src in disCountsBySeconds[second]:
    disSrcCount = disCountsBySeconds[second][src]
else:
    disSrcCount = 0
if len(dioCountsBySeconds[second]) > 0 and src in dioCountsBySeconds[second]:
    dioSrcCount = dioCountsBySeconds[second][src]
else:
    dioSrcCount = 0

```



```
dao = daoSrcCount
dis = disSrcCount
dio = dioSrcCount
label = 1
row = np.append(row, str(transmissionRate))
row = np.append(row, str(receptionRate))
row = np.append(row, str(TrRr))
row = np.append(row, str(srcCount))
row = np.append(row, str(dstCount))
row = np.append(row, str(trnTtlDur))
row = np.append(row, str(rcvTtlDur))
row = np.append(row, str(trnAverageTime))
row = np.append(row, str(rcvAverageTime))
row = np.append(row, str(dao))
row = np.append(row, str(dis))
row = np.append(row, str(dio))
row = np.append(row, str(label))
monitorArray.append(row)
```

```
counter += 1
```

```
headers = [
    'No.', 'Time', 'Source', 'Destination', 'Length', 'Info',
    'Transmission Rate (per 1000 ms)',
    'Reception Rate (per 1000 ms)',
    'TR / RR',
    'Sources Count Per Sec',
    'Destinations Count Per Sec',
    'Trans Total Duration Per Sec',
```

```
'Rcv Total Duration Per Sec',
'Trans Average Per Sec',
'Rcv Average Per Sec',
'DAO',
'DIS',
'DIO',
'Label'
]
monitorArray.insert(0, headers)

with open('forward-blackhole-result.csv', 'w') as monitoring:
    wr = csv.writer(monitoring, dialect='excel', delimiter=',')
    wr.writerows(monitorArray)
#####

#####
# Script 2 - Data Normalization Algorithm #
#####

# import the required packages
import pandas as pd
...

# dataset headers
path1='./csvs/'
headers_val = [
    'Length', 'Info',
    'Transmission Rate (per 1000 ms)',
    'Reception Rate (per 1000 ms)',
    'TR / RR',
    'Sources Count Per Sec',
    'Destinations Count Per Sec',
```

```
'Trans Total Duration Per Sec',
'Rcv Total Duration Per Sec',
'Trans Average Per Sec',
'Rcv Average Per Sec',
'DAO',
'DIS',
'DIO'
]
amgPd = pd.DataFrame()

# import dataset
for chunk in pd.read_csv(path1+'BH1000%5mix.csv', chunksize = 250000, low_memory=False):
    amgPd = pd.concat([amgPd,chunk])
print ('BH1000%5mix')
print (amgPd.describe())

# transform matrix format
A = amgPd.as_matrix()
row_num = A.shape[1]
X = A[:, 0:row_num - 1]
Y = A[:, row_num - 1].astype(int)

# feature normalization #

X = preprocessing.quantile_transform (X, output_distribution='normal')
X = preprocessing.minmax_scale (X)
data = pd.DataFrame(X, columns=headers_val)
labels = pd.DataFrame(Y)

# creating main dataset #
```

```
amgPd = data.join(labels.rename(columns=0:'Label'))
print ('BH1000%5mix after scaling')
print (amgPd.describe())

# transform to csv of normalized dataset
amgPd.to_csv('BH1000%5mix_norm.csv', index=False)

# import another dataset
amgPd1 = pd.DataFrame()
for chunk in pd.read_csv(path1+'BH100%10mix.csv', chunksize = 250000, low_memory=False):
    amgPd1 = pd.concat([amgPd1,chunk])
amgPd1.drop([u'No.', 'Time', u'Source', u'Destination'], axis=1, inplace=True)
print ('BH100%10')
print (amgPd1.describe())

# transform matrix format
A = amgPd1.as_matrix()
row_num = A.shape[1]
X = A[:, 0:row_num - 1]
Y = A[:, row_num - 1].astype(int)

# feature normalization #

X = preprocessing.quantile_transform (X, output_distribution='normal')
X = preprocessing.minmax_scale (X)
data = pd.DataFrame(X, columns=headers_val)
labels = pd.DataFrame(Y)

# concatenating the datasets #

amgPd1 = data.join(labels.rename(columns=0:'Label'))
print ('BH100%10 after scaling')
```

```
print (amgPd1.describe())
amgPd1.to_csv('BH100%10mix_norm.csv', index=False)

# appending to main dataset
amgPd = pd.concat ([amgPd, amgPd1])

# Another datasets are applied same process #

#####
# Script 3 - Deep Learning Algorithm #
#####

# import the required packages
import csv
...

# import dataset
dftrain = pd.read_csv('BHdataset.csv', encoding='utf-8-sig')

# data shuffling
dftrain = shuffle(dftrain)

#feature importance
feat_labels = dftrain.columns[0:]
forest = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1,verbose=1)
forest.fit(X, Y)
importances = forest.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(X.shape[1]):
    print ("%2d %-*s %f" % (f + 1, 30, feat_labels[f], importances[indices[f]]))
```

```
#feature selection
dftrain.drop([], axis=1, inplace=True)
A = dftrain.as_matrix()
row_num = A.shape[1]
X = A[:, 0:row_num - 1]
Y = A[:, row_num - 1].astype(int)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1)

# creating deep learning model
model = Sequential()

model.add(Dense(kernel_initializer="uniform", activation="relu", input_dim=10, units=20,
                bias_regularizer=regularizers.l1(0.005)
                )
)

# setting deep layers
model.add(Dense(50, kernel_initializer="uniform", activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Dense(100, kernel_initializer="uniform", activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Dense(300, kernel_initializer="uniform", activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Dense(100, kernel_initializer="uniform", activation="relu"))
model.add(BatchNormalization())
```

```
model.add(Dropout(0.1))

model.add(Dense(50, kernel_initializer="uniform", activation="relu"))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Dense(1, init="uniform"))
model.add(Activation('sigmoid'))

adagrad = optimizers.Adagrad(lr=0.05, epsilon=1e-08, decay=0.0)

model.compile(loss='binary_crossentropy',
              optimizer=adagrad,
              metrics=["accuracy"])
# model training starts
history = model.fit(X_train,
                   Y_train,
                   nb_epoch=1000,
                   batch_size=250,
                   validation_split=0.3,
                   verbose=2,
                   class_weight=class_weight)

scores = model.evaluate(X_test, Y_test)

# classification report
probabilities = model.predict(X_test)
predictions = [float(x.round()) for x in probabilities]
cr = classification_report(Y_test,predictions)
print(cr)
```

```
# saving deep learning model
model_json = model.to_json()
with open("model_bh.json", "w") as json_file:
    json_file.write(model_json)

model.save_weights("model_bh.h5")
print("Saved model to disk")

# training accuracy
print(history.history.keys()) plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# training loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
gc.collect()
```

```
#####
```


Bibliography

- [1] W. Wahlster. From industry 1.0 to industry 4.0: Towards the 4th industrial revolution. In *Forum Business meets Research*, 2012.
- [2] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local computer networks, proceedings 2006 31st IEEE conference on*, pages 641–648. IEEE, 2006.
- [3] Gns-3, 2017. URL <https://www.gns3.com/>.
- [4] Iotify, 2017. URL <https://iotify.io/iot-network-simulator/>.
- [5] Matlab, 2017. URL <https://www.mathworks.com/solutions/internet-of-things.html>.
- [6] Ddos attacks increased 91 URL <https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/>.
- [7] OWASP. Top iot vulnerabilities. URL https://www.owasp.org/index.php/Top_IoT_Vulnerabilities.
- [8] Symantec. Internet security threat report. Technical report, Volume:22, April 2017.
- [9] Kaspersky. Black hat usa 2015: The full story of how that jeep was hacked. URL <https://www.kaspersky.com/blog/blackhat-jeep-cherokee-hack-explained/9493/>.
- [10] A. A. Ghorbani, W. Lu, and M. Tavallae. *Network intrusion detection and prevention: concepts and techniques*, volume 47. Springer Science & Business Media, 2009.
- [11] D. Janakiram, V. Reddy, and A. P. Kumar. Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and*

- Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–6. IEEE, 2006.
- [12] S. Mascaro, A. E. Nicholso, and K. B. Korb. Anomaly detection in vessel tracks using bayesian networks. *International Journal of Approximate Reasoning*, 55(1): 84–98, 2014.
- [13] Y. Zhang, N. Meratnia, and P. J. Havinga. Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine. *Ad hoc networks*, 11(3):1062–1074, 2013.
- [14] S. Kaplantzis, A. Shilton, N. Mani, and Y. A. Sekercioglu. Detecting selective forwarding attacks in wireless sensor networks using support vector machines. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 335–340. IEEE, 2007.
- [15] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 3864–3869. IEEE, 2007.
- [16] Z. Yang, N. Meratnia, and P. Havinga. An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine. In *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*, pages 151–156. IEEE, 2008.
- [17] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.
- [18] T. Hurley, J. E. Perdomo, and A. Perez-Pons. Hmm-based intrusion detection system for software defined networking. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pages 617–621. IEEE, 2016.
- [19] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.
- [20] M. N. Chowdhury, K. Ferens, and M. Ferens. Network intrusion detection using machine learning. In *Proceedings of the International Conference on Security and Management (SAM)*, page 30. The Steering Committee of The World Congress in

- Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016.
- [21] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.
- [22] L. Wallgren, S. Raza, and T. Voigt. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, 9(8):794326, 2013.
- [23] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito. An ids framework for internet of things empowered by 6lowpan. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1337–1340. ACM, 2013.
- [24] P. Pongle and G. Chavan. Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 121(9), 2015.
- [25] F. Nait-Abdesselam, B. Bensaou, and T. Taleb. Detecting and avoiding wormhole attacks in wireless ad hoc networks. *IEEE Communications Magazine*, 46(4):127–133, 2008.
- [26] Z. Banković, D. Fraga, J. M. Moya, and J. C. Vallejo. Detecting unknown attacks in wireless sensor networks that contain mobile nodes. *Sensors*, 12(8):10834–10850, 2012.
- [27] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [28] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [29] CompTIA. Sizing up the internet of things, Lastchecked: 15/11/2017. URL <https://www.comptia.org/images/default-source/Insight-Tools/Thumbnails/internetofthings.png?sfvrsn=0>.
- [30] L. Columbus. 2017 roundup of internet of things forecasts. URL <https://www.forbes.com/sites/louiscolombus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#1ba8b8651480>.

- [31] S. Sargolzaei, M. Cabrerizo, A. Sargolzaei, S. Noei, and M. Adjouadi. Epilepsy, a cyberattack on brains' networked control system. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pages 622–625. IEEE, 2016.
- [32] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-physical systems security—a survey. *IEEE Internet of Things Journal*, 2017.
- [33] B. Zhu, A. Joseph, and S. Sastry. A taxonomy of cyber attacks on scada systems. In *Internet of things (iThings/CPSCoM), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.
- [34] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.
- [35] R. Roman, P. Najera, and J. Lopez. Securing the internet of things. *Computer*, 44(9):51–58, 2011.
- [36] J. A. M. M. Jazib Frahim, Carlos Pignataro. Securing the internet of things: A proposed framework. CISCO, 2016. URL <https://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html>.
- [37] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.
- [38] K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE wireless communications*, 11(6):54–61, 2004.
- [39] M. A. M. Vieira, C. N. Coelho, D. Da Silva, and J. M. da Mata. Survey on wireless sensor network devices. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, volume 1, pages 537–544. IEEE, 2003.
- [40] E. H. Callaway Jr. *Wireless sensor networks: architectures and protocols*. CRC press, 2003.
- [41] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, 30(7):1655–1695, 2007.

- [42] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, 2004.
- [43] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.
- [44] F. Gont. Results of a security assessment of the internet protocol version 6 (ipv6).
- [45] M. Abomhara et al. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility*, 4(1):65–88, 2015.
- [46] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.
- [47] T. Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.
- [48] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.
- [49] A. Mayzaud, R. Badonnel, and I. Chrisment. A taxonomy of attacks in rpl-based internet of things. *International Journal of Network Security*, 18(3):459–473, 2016.
- [50] A. Aris, S. F. Oktug, and S. B. O. Yalcin. Rpl version number attacks: In-depth study. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/I-FIP*, pages 776–779. IEEE, 2016.
- [51] M. A. Maloof. *Machine learning and data mining for computer security: methods and applications*. Springer, 2006.
- [52] E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.
- [53] A. A. Diro and N. Chilamkurti. Distributed attack detection scheme using deep learning approach for internet of things. *Future Generation Computer Systems*, 2017.
- [54] L. Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.

- [55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- [56] R. DiPietro. A friendly introduction to cross-entropy loss. URL <https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/>.
- [57] A. Garofalo, C. Di Sarno, and V. Formicola. Enhancing intrusion detection in wireless sensor networks through decision trees. In *Dependable Computing*, pages 1–15. Springer, 2013.
- [58] G. Kibirige and C. Sanga. A survey on detection of sinkhole attack in wireless sensor network. 13:1–9, 05 2015.
- [59] S. Sharma and A. Nayyar. Mint-route to avoid congestion in wireless sensor network. *International Journal of Emerging Trends & Technology in Computer Science*, 3(2): 91–94 March, 2014.
- [60] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos. Intrusion detection of sinkhole attacks in wireless sensor networks. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 150–161. Springer, 2007.
- [61] L. Coppolino, S. D’Antonio, L. Romano, and G. Spagnuolo. An intrusion detection system for critical information infrastructures using wireless sensor network technologies. In *Critical Infrastructure (CRIS), 2010 5th International Conference on*, pages 1–8. IEEE, 2010.
- [62] M. Strohmeier, M. Smith, M. Schäfer, V. Lenders, and I. Martinovic. Crowdsourcing security for wireless air traffic communications. In *Proceeding of the 9th International Conference on Cyber Conflict (CYCON)*. IEEE, jun 2017. URL http://www.cs.ox.ac.uk/files/9099/CyCon_2017_Strohmeier_Smith_Schaefer_Lenders_Martinovic.pdf.
- [63] <https://opensky.network.org>. Opensky.
- [64] U. S. R. K. Dhamodharan and R. Vayanaperumal. Detecting and preventing sybil attacks in wireless sensor networks using message authentication and passing method.

- The Scientific World Journal*, 2015:7, 2015. URL <http://dx.doi.org/10.1155/2015/841267>].841267.
- [65] R. Kaur and M. Singh. Efficient hybrid technique for detecting zero-day polymorphic worms. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 95–100. IEEE, 2014.
- [66] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *AAAI/IAAI*, pages 217–223, 2002.
- [67] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering*, 21(3):181–199, 1995.
- [68] Y. Alofer and O. F. Rana. Predicting client-side attacks via behaviour analysis using honeypot data. In *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, pages 31–36. IEEE, 2011.
- [69] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg. Rule-based anomaly detection on ip flows. In *INFOCOM 2009, IEEE*, pages 424–432. IEEE, 2009.
- [70] S. Raza, L. Wallgren, and T. Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8):2661–2674, 2013.
- [71] T. Avram, S. Oh, and S. Hariri. Analyzing attacks in wireless ad hoc network with self-organizing maps. In *Communication Networks and Services Research, 2007. CNSR'07. Fifth Annual Conference on*, pages 166–175. IEEE, 2007.
- [72] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). Technical report, 2003.
- [73] T. Issariyakul and E. Hossain. *Introduction to network simulator NS2*. Springer Science & Business Media, 2011.
- [74] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. *Knowledge and information systems*, 34(1):23–54, 2013.
- [75] C. Koliass, G. Kambourakis, A. Stavrou, and S. Gritzalis. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2016.

- [76] J. Ryan, M.-J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. In *Advances in neural information processing systems*, pages 943–949, 1998.
- [77] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles. Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.
- [78] A. Javaid, Q. Niyaz, W. Sun, and M. Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONET-ICS)*, pages 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.
- [79] Y. Li, R. Ma, and R. Jiao. A hybrid malicious code detection method based on deep learning. *methods*, 9(5), 2015.
- [80] K. J.-W. Kang M.-J. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE 11(6): e0155781. doi:10.1371/journal.pone.0155781*, 2016.
- [81] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE, 2009.
- [82] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. SciPy Austin, TX, 2010.
- [83] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.
- [84] CETIC. Foren6. URL <http://cetic.github.io/foren6/>.
- [85] N. Moustafa, J. Slay, and G. Creech. Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactions on Big Data*, PP(99):1–1, 2017. doi: 10.1109/TBDDATA.2017.2715166.

- [86] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [87] F. Chollet. Keras. <https://keras.io>.
- [88] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [89] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [90] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [91] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *Military Communications and Information Systems Conference (MilCIS), 2015*, pages 1–6. IEEE, 2015.