UNIVERSITY OF BAHRAIN

College of
Information Technology

Department of
Computer Science

Performance of Machine Learning for Anomaly Detection in Dynamic Network Graphs

A Thesis Submitted in Partial Fulfilment of the Requirements for the M.Sc. Degree in Cyber Security

Submitted by
Sara Saad Alromaihi
ID Number: 20122438

Supervised by
Dr. Hesham Mohammed Al-Ammal
(Assistant Professor)
University of Bahrain

Kingdom of Bahrain
January 2022

University of Bahrain
College of Information Technology



Performance of Machine Learning for Anomaly Detection in Dynamic Network Graphs

A Thesis Submitted in Partial Fulfilment of the Requirements for the degree of M.Sc. in Cyber Security

Submitted By:

Student's Name: Sara Alromaihi

ID Number: 20122438

Supervised By:

Dr. Hesham Al-Ammal

(Assistant Professor)

University of Bahrain

Kingdom of Bahrain

January 29, 2022

University of Bahrain
Deanship of Graduate Studies
And scientific Research



جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Dissertation Deposit Form

Author Name:	Sara Saad Alromaihi		
Email:	sara.alrumaihi@gmail.com	Contact No:	36936328
College:	**Information Technology**		
Department:	**Computer Science**		
Thesis Title:	Performance of Machine Learning for Anomaly Detection in Dynamic Network Graphs		
Year:	2022		

Author's Declaration: I agree to the following conditions:

The above thesis will be available (in the University of Bahrain library and externally) and reproduced as necessary at the discretion of the University of Bahrain. It may also be digitized by the University of Bahrain and made available via the university's repository or via other parties on the Internet.

This condition shall apply to ALL copies including electronic copies.

The above thesis has been provided on the understanding that it is copyright protected material and that no quotation from this thesis may be published without proper acknowledgement.

Signature of the Author:	Sara Alromaihi	Date:	29-Jan-22

Approved at the University Council 6th meeting, 2013
In accordance with Decision No. 736/2013 dated 14/3/2013
Modified by University Council Decision No. 2043/2013 dated 20/11/2013.

DECLARATION

I declare that this work is the result of my own investigation and that it has not already been accepted in substance for any degree, nor is it currently submitted for any degree.

Signed:  Date: 29/01/2022

Sara Saad Alromaihi (Candidate)

The Thesis defense committee considers the thesis titled:

Performance of Machine Learning for Anomaly Detection in Dynamic Network Graphs

Submitted by Sara Saad Alromaihi to the College of Information Technology is satisfactory and acceptable for the Master of Science in Cyber Security.

The Defense Committee:

Dr. Hesham Al-Ammal (Supervisor) — Department of Computer Science, University of Bahrain

---

Dr. Mhd Saeed Sharif (External Examiner) — University of East London (UEL)

Prof. Faisal Alkhateeb (Internal Examiner) — Department of Computer Science, University of Bahrain

Acknowledgements

This thesis became a reality with the kind support and help of many individuals. I would like to extend my gratitude to all of them.

First, I would like to convey my sincere appreciation to my supervisor, Dr. Hesham Al-Ammal, whose guidance not only made me a better researcher, but also a better person.

I would also like to thank my family for their continuous support. My gratitude specially goes to my beloved mother for all the unconditional love, guidance, and support that she has given me; helping me to succeed and instilling in me the confidence that I am capable of doing anything I put my mind to.

A very special thanks goes to my late brother, Mubarak, for teaching me the true meaning of hard-work and sacrifice.

Abstract

Detection of anomalies in a given dataset is a key step in several applications in cybersecurity; including intrusion detection, fraud, and social network analysis. Many of these methods detect anomalies by analyzing graph-based data. Examining graphs makes it possible to capture relationships, communities, as well as anomalies. The advantage of using graphs is that many real-life situations can be easily modeled by a graph that captures their structure and inter-dependencies. Although anomaly detection in graphs dates back to 1990s, recent advances in research utilized machine learning methods for anomaly detection over graphs. The thesis aims at investigating several deep learning algorithms for the discovery of anomalous behavior in network activity. The aim is to understand under which conditions each algorithm performs better and compare their efficiency and accuracy for well-known datasets. All models were trained and tested on the CIC-IDS2018 dataset that was created by The Canadian Institute for Cybersecurity. It was found that the achieved accuracy is 99.60% using feature-based anomaly detection. Although most related work neglected the AUC metric, which relates to the disproportional class distribution of the dataset, the AUC metric is considered the most robust performance metric, and in our work, the top possible scores of AUC are achieved.

Contents

ACKNOWLEDGEMENTS i
ABSTRACT ii
TABLE OF CONTENTS iii
LIST OF TABLES v
LIST OF FIGURES vi
ACRONYMS vii

1 Introduction **1**
1.1 Overview 2
1.2 Background 3
1.2.1 Graphs 3
1.2.2 Anomaly Detection 5
1.2.3 Machine Learning 6
1.3 Problem Definition 7
1.4 Research Objectives 9
1.5 Research Significance 9
1.6 Thesis Organization 10

2 Literature Review **11**
2.1 Anomaly Detection in Static Network Graphs 11
2.1.1 Structure-based Anomaly Detection 11
2.1.2 Community-based Anomaly Detection 13
2.2 Anomaly Detection in Dynamic Network Graphs 14
2.2.1 Anomalous node detection 14
2.2.2 Anomalous edge detection 16
2.2.3 Anomalous subgraph detection 17
2.3 Deep Learning on Graphs for Anomaly Detection 18

2.3.1 Node Embedding 18
2.3.2 Inductive Representation Learning 19
2.3.3 Anomaly Detection over Graph Embedding 20

3 Methodology **21**
3.1 Feature-based Anomaly Detection 21
3.1.1 Motivation 21
3.1.2 Graph Transform 22
3.1.3 Feature Extraction 23
3.1.4 Feature Normalization 27
3.1.5 Time Series Conversion 27
3.1.6 Long Short-Term Memory Model 28
3.2 Anomaly Detection using Node Embeddings 28
3.2.1 Temporal Random Walk 28
3.2.2 node2vec 29
3.2.3 Logistic Regression 30
3.3 Summary 30

4 Results and Discussion **31**
4.1 Dataset 31
4.2 Performance Metrics 32
4.3 Results 33
4.3.1 Performance of Feature-based Anomaly Detection 33
4.3.2 Performance of Anomaly Detection using Node Embeddings 36
4.4 Discussion 38

5 Conclusion and Recommendations **41**
5.1 Conclusion 41
5.2 Research Limitations and Future Work 43
Bibliography 44
ملخص 54

List of Tables

3.1 Sample of extracted time-related features in the CIC-IDS2018 dataset 26

4.1 Specifications of CIC-IDS2018 Dataset 32
4.2 Details of the selected attack categories 32
4.3 Performance of the stateless LSTM on the analyzed attack categories 34
4.4 Performance of the stateful LSTM on the analyzed attack categories 34
4.5 Performance of the anomaly classifier based on node embeddings 38
4.6 Comparison of performance scores with similar work based on CIC-IDS2018 39

List of Figures

1.1 Number of Internet users worldwide from 2017 to 2025 (Statista, 2020) . . . 2
1.2 Percentage of HTTPS usage by year (Google Transparency Report, 2020) . . 3
1.3 Node ordering based on the selected graph algorithm 5

2.1 An example of anomalous node detection based on community changes . . . 15
2.2 An example of anomalous edges based on abnormal weight evolution 16
2.3 Two examples of anomalous subgraphs 17

3.1 Components of the anomaly detection system 22
3.2 A sequence of graph snapshots generated from the packet capture file of the botnet category 23
3.3 An example of a TTL expiry attack 26
3.4 The overall traffic in the capture file measured in bytes per minute 27
3.5 Components of the anomaly detection system 29

4.1 Scatter plot of imbalanced dataset before and after undersampling 32
4.2 ROC curve of the stateless model trained on the analyzed attacks 35
4.3 ROC curve of the stateful model trained on the analyzed attacks 36
4.4 A tSNE visualization of the node embedding for the hosts 37
4.5 A visualization of the classification based on node embedding 38

Acronyms

DOS	Denial of Service
SSL	Secure Socket Layer
DFS	Depth First Search
BFS	Breadth First Search
IDS	Intrusion Detection System
P2P	Peer-to-peer
C&C	Command & Control
ML	Machine Learning
DL	Deep Learning
CNN	Convloutional Neural Network
GCN	Graph Convloutional Network
NLP	Natural Language Processing
NNMF	Non-negative matrix factorization
TTL	Time-to-live
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
API	Application Programming Interface
ADAM	Adaptive Moment Estimation
ROC	Receiver Operating Characteristics
AUR	Area Under Receiver Operating Characteristics Curve

Chapter 1

Introduction

The human brain has subconscious but sophisticated machinery for identifying abnormalities from normality and responding to them in a quick manner. Kahneman (2011) describes this phenomena:

> Studies of brain responses have shown that violations of normality are detected with astonishing speed and subtlety. In a recent experiment, people heard the sentence "Earth revolves around the trouble every year." A distinctive pattern was detected in brain activity, starting within two-tenths of a second of the onset of the odd word.... A vast amount of world knowledge must instantly be brought to bear for the incongruity to be recognized. (Kahneman, 2011)

The early detection of abnormal events is as valuable to machines as it is to humans. With extremely large amounts of data arriving in a constant manner, the machine (or *algorithm*) has to act as a human brain; to rapidly identify and respond to anomalies. Therefore, with the increasing availability of data, anomaly detection has seen increasing attention by researchers. Based on Statista (2020), one of the world's largest online statistics database, there are around 4 billion social media users around the world in the year 2020, a number expected to increase to almost 4.5 billion in the next five years (see Figure 1.1).

The availability of these datasets is increasing the need for *cognitive* algorithms that can automatically identify anomalous users or events in a quick and reliable fashion. In the cybersecurity domain, we are mainly interested in correctly detecting potentially malicious hosts and intrusions (e.g., botnet). These attacks are typically launched from one or more computers against other computers or networks to either disable the target computer or gain access to its data.

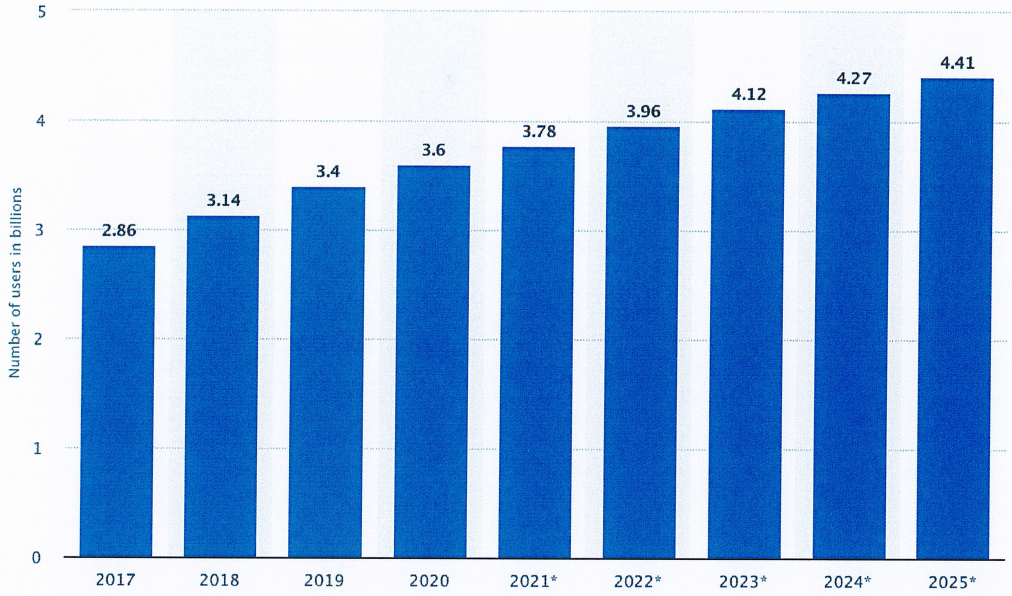


Figure 1.1: Number of Internet users worldwide from 2017 to 2025 (Statista, 2020)

1.1 Overview

The cybersecurity threat landscape is constantly evolving. In this interconnected world, a breach on one company can compromise an entire supply chain. A recently disclosed data wiping malware called Dustman has been launched against energy companies in the Middle East region. In May 2017, the WannaCry ransomware had a substantial impact across the globe with more than $4 billion in losses. In addition, the 2016 Dyn cyberattack has turned IoT devices into a huge botnet that caused significant outages to major service providers in Europe and North America.

More sophisticated cyber attacks have emerged recently targeting educational, financial, and governmental sectors (Kopp et al., 2017; Singar and Akhilesh, 2020; Langner, 2011). Attackers use advanced tools to break into the target's network and maintain the connection undetectable. In addition, many samples of malware are specially crafted to evade the security tools (e.g., anti-virus and firewalls) as they usually fail in detecting malicious activities at an early stage. The advantages that come with secure sockets layer (SSL) encrypted traffic can also be used by cyber-criminals to conceal and launch encrypted malware attacks. When a malicious payload is concealed in encrypted traffic, it becomes almost impossible to detect it. According to Google Transparency Report (2020), nearly 80% of all web pages now use SSL

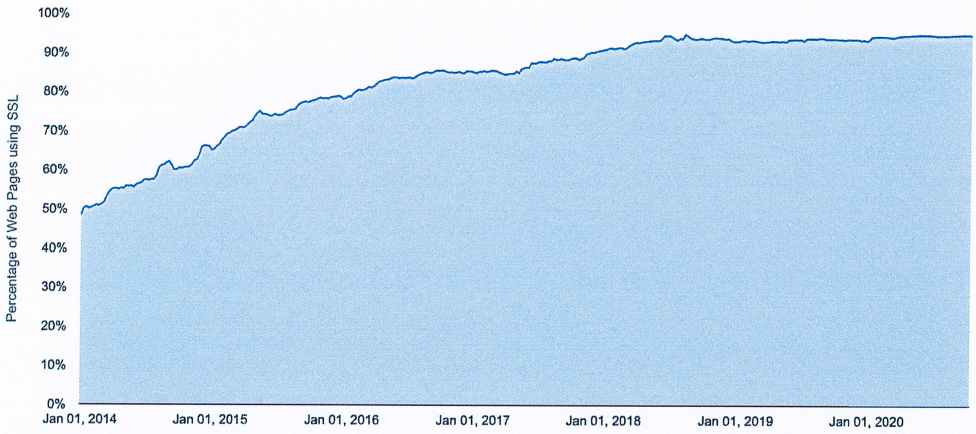


Figure 1.2: Percentage of HTTPS usage by year (Google Transparency Report, 2020)

and over 50% of today's malware is hiding in SSL (see Figure 1.2). Many existing anomaly detection methods rely on deep packet inspection where the full content of the packet (i.e., the payload) is examined. Such methods are unreliable since attackers can evade detection by using encryption or changing the packet structure.

Graph analysis overcomes this deficiency as graph-based features can reveal meaningful patterns in the links established with other nodes. It is also more robust than flow-based methods because it does not require to cross compare traffic in the dataset (Venkatesh et al., 2015). Thus, in order to prevent these attacks, detecting anomalous behaviors as accurately and early as possible becomes inevitable.

1.2 Background

In this section, we introduce the fundamental definitions in graph theory, anomaly detection, and machine learning (ML).

1.2.1 Graphs

Graphs are extensively used to capture relationships and interactions between different entities in many domains such as social networks, sociology, and chemistry. We begin with basic graph definitions and characteristics followed by a summary of graph algorithms.

Definition 1.2.1. In computer science, a *graph* (or *network*) is a data structure that models pairwise communication between some objects. It consists of a set of *nodes* (or *vertices*), and

a set of *edges* (or *links*) between pairs of nodes. For instance, nodes could represent users, and edges could represent collaborations between those users. Mathematically, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges.

Definition 1.2.2. A *weighted graph* $\mathcal{G} = (\mathcal{V}, E, w)$ has an associated weight function $w : \mathcal{E} \rightarrow \mathbb{R}$. It is a graph in which each edge is given a numerical weight, which might represent costs, ratings or lengths, depending on the problem at hand.

Definition 1.2.3. A *bipartite graph* divides the set of nodes into two sets, $\mathcal{U}$ and $\mathcal{N}$, where all edges connect a node in $\mathcal{U}$ and a node in $\mathcal{N}$. For example, given a media-services provider (e.g., Netflix), $\mathcal{U}$ represents the set of viewers, and $\mathcal{N}$ represents the set of movies, and an edge exists from u to n if u played the movie m.

Definition 1.2.4. A *subgraph* is a subset of the graph whose set of nodes and set of edges are all subsets of $\mathcal{G}$.

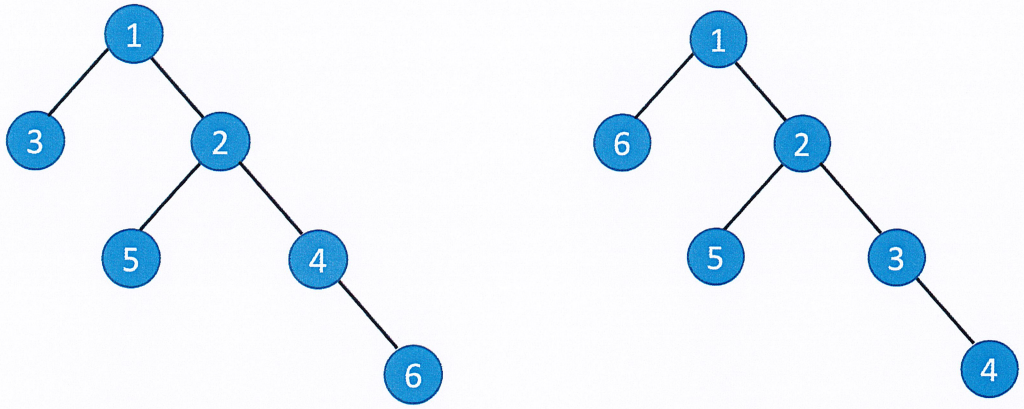
Definition 1.2.5. Two nodes are considered *neighbors* when they are connected by an edge.

Definition 1.2.6. The *neighbourhood* of a node v in a graph $\mathcal{G}$ is the subgraph of $\mathcal{G}$ induced by all nodes adjacent to n (i.e., the node and all its neighbours).

Definition 1.2.7. The *node degree* is the total number of neighbours the node has.

There are two ways of traversing a graph: depth first search (DFS) and breadth first search (BFS). The aim of these algorithms is to specify an order to search through the nodes of a graph (see Figure 1.3). In depth first search the algorithm starts at a node and explores as far as possible along a path away from the starting node. In breadth first search, the algorithm explores all of the neighbor nodes at the present depth, then moving to the neighbors of neighbors, and so on.

The aforementioned graph characteristics facilitate the anomaly detection task. This is due to the possibility of modeling many real-life scenarios with graphs that capture their interactions and dependencies. Interactions on networks generally have three dimensions; a content dimension, a social dimension, and a temporal dimension. The first dimension describes the correlation among packet payloads. The social dimension involves the relations among nodes (e.g.,, users or servers). Finally, the temporal dimension describes the change of users' behaviors over time. These relations are useful for detecting anomalies in the network.



(a) Depth First Search (DFS)

(b) Breadth First Search (BFS)

Figure 1.3: Node ordering based on the selected graph algorithm

1.2.2 Anomaly Detection

The ubiquitousness of accurate Intrusion Detection Systems (IDSs) is growing significantly as the threat landscape changes. Intrusion detection systems identify suspicious activity and known threats such as reconnaissance, enumeration, port scanning, and flooding attacks. Generally, IDSs are classified as host-based or network-based.

A host-based IDS runs on an endpoint device. It monitors incoming and outgoing traffic from the device only and alerts the user on the detection of anomalous activity. This detection method is best suited to devices that are not expected to modify their configurations. On the other hand, a network-based IDS observes incoming and outgoing traffic from all devices connected to the network. However, scanning all incoming and outgoing traffic can cause a bottleneck and reduce the throughput of the network.

Intrusion detection systems are also categorized into either signature-based or anomaly-based. A signature-based IDS is modeled using specific attack patterns, such as known malicious byte sequence of malware. Although signature-based systems are able to detect known attacks with high accuracy, they fail in detecting novel attacks in which no specific pattern exists. In addition, the security provided by traffic encryption is being misused by attackers to hide malicious content, such as malware payloads and command and control (C&C) activities. Needless to say, malicious payloads are crafted to silently exist for extended periods while communicating via encryption with the botmaster and its C&C channels.

An anomaly-based intrusion detection system is an important approach in network security. After defining a set of ordinary profiles, any event that falls outside the predefined

profile raises a flag (Hawkins, 1980). These irregular patterns are usually referred to as anomalies or outliers. Anomaly-based systems allow security analysts to effectively identify network intrusions and maximize security through the early detection of unusual events in the network.

However, these systems require monitoring critical network attributes in real-time to produce an alarm if abnormal trends are detected. In particular, it is difficult to detect relationships or points in time that significantly deviate from the rest to identify abnormal users or events in the network given the streaming nature of data. Therefore, anomaly detection is an essential problem in dynamically changing networks. With the rapid advancements in various machine learning frameworks, and due to their great success in many applications, anomaly detection using machine learning has become more interesting. Machine learning models can be used to accurately identify anomalies after being trained on a relevant dataset.

1.2.3 Machine Learning

Machine learning involves the use of statistical techniques for enhancing the performance of a computer system without being explicitly programmed. The term was first introduced in 1959, inspired from related fields such as computational learning theory and pattern recognition (Samuel, 1959). Machine learning is a method used to make decisions based on learning from relationships in data. The tasks of machine learning are classified into two general groups, one is *supervised learning* and the other is *unsupervised learning*. In a supervised learning, object labels are given to the learning algorithm and the aim is to map inputs to outputs based on example pairs. On the contrary, in an unsupervised learning, the algorithm has no previous knowledge of data labels (Murphy, 2013).

Supervised learning uses algorithms to learn the mapping function from an input to an output based on training pairs. The mapping function is derived from a labeled training object which consist of multiple training examples. The aim of supervised learning is to approximate the mapping function and predict the output result as accurately as possible. The algorithm iteratively produces predictions on the training dataset and stops when an acceptable level of performance is achieved. Furthermore, supervised learning problems can be categorized as regression and classification. When the output is a category (e.g., malicious or benign), it is considered as classification problem. Conversely, when the output data is a real value (e.g., weight), it is considered as a regression problem.

The aim of unsupervised learning is to identify the underlying structure of a dataset to effectively represent data in a grouped (clustering) or compressed (reduced dimensionality)

format. This type of learning is unsupervised as no labeled data exist. Unsupervised learning techniques are candidates for anomaly detection tasks. Although they have been successfully implemented in host-based intrusion detection systems (Kruegel et al., 2003), their application to network based systems is not straightforward. This is due to the challenges of input selection, data dimensionality and throughput.

Recently, machine learning research have received renewed attention fostered by the advances in representation learning and deep learning. Deep learning has been considered as the "crown jewel" of artificial intelligence (LeCun et al., 2015), achieving outstanding performance in speech recognition (Hinton et al., 2012), computer vision (Krizhevsky et al., 2012), and natural language processing (Bahdanau et al., 2014), etc. Geometric deep learning, a branch of deep learning that deals with data defined in non-Euclidean space (i.e., graphs and manifolds), is a promising tool to solve graph related tasks in many applications such as recommendation systems, molecular interactions, and genomics (Hamilton et al., 2017b; Bronstein et al., 2017; Duvenaud et al., 2015; Dai et al., 2016; Gilmer et al., 2017; Kipf and Welling, 2016). Despite the significant progress achieved, extracting graph-level features that can be exploited by machine learning algorithms remains a challenging problem, especially for complex, time-evolving networks.

1.3 Problem Definition

A graph is a powerful representation of objects and the relationships among them. For instance, graphs can represent financial systems that connect financial institutions globally. They can also represent social networks connecting users through relationships such as friendship and/or collaboration.

However, the detection of anomalous behaviors is challenging. First, it is difficult to monitor and process dynamic data that changes over time. In addition, most real-world data has a streaming nature that demands real-time processing. It is also not straightforward to define a boundary between normal and abnormal activity. Generally, anomaly-based detection systems may generate false alarms when an unusual legitimate event is classified malicious, known as *false positive*. Moreover, attackers usually adjust malicious codes to ordinary patterns. Hence, such attacks have a high probability of being classified as ordinary behavior, resulting in *false negatives*. Therefore, it is important to mitigate false positives and false negatives for building an effective intrusion detection system. It is also important to take into account the dynamic nature of network data, which introduces new challenges,

such as:

Various types of anomalies. Since dynamic graphs evolve over time, anomalies can take many different forms. Possible types of anomalies in evolutionary networks include splitting, merging, disappearing, growing, and shrinking communities.

Graph size. In the big-data era, most graphs (e.g., social network or e-commerce) can have billions of objects that are inserted and discarded over time. Those graph updates need to be stored and processed after the dynamic changes. However, storing every graph in its entirety is simply too large. Also, typical offline analysis becomes infeasible as it requires multiple passes over the data. Similarly, using only the latest graph limits the analysis to a single point in time.

Graph structure. Unlike images and text, graphs have varying structures that it becomes difficult to generalize some of the basic operations to graphs (Shuman et al., 2013). Nodes have arbitrary locations on the graph and two connected nodes are not necessarily "close". Since the structure of the graph may constantly change over time, the underlying classification model may also change. Therefore, the model has to be efficiently updatable in order to account for such changes. In addition, different portions of the graph may be better suited to different kinds of models. For instance, some portions of the network may be better classified with structure, whereas other portions may be better classified with context. Due to the irregular structure of graphs, it is very complex to define convolution and pooling operations for graph data, some basic operations in convolutional neural networks (CNNs). This is known as the *geometric deep learning problem* (Bronstein et al., 2017).

Representation learning. It is difficult to encode graph structure in a way that can be leveraged by ML models. More importantly, as more data are generated, the question becomes 'what useful trends could be extracted from these streams to enhance the detection accuracy?'.

Dynamicity. Many such networks are updated over time. Thus, it is difficult to detect points in time that significantly deviate from the rest to identify anomalous behaviors.

Diversity of graphs. A graph can include many types and properties of nodes and edges. For instance, graphs can be directed or undirected, homogeneous or heterogeneous, and weighted or unweighted. Their tasks also vary significantly, ranging from node-oriented problems such as link prediction to graph-oriented tasks such as graph classification. These different forms, properties, and tasks require different models to tackle them accordingly.

Domain knowledge. Graphs are typically related to other domains, social networks in computing, financial networks in banking, anatomical networks in neuroscience, and protein

networks in biology. Although domain knowledge is important in solving specific tasks, it can complicate the design of the model. In molecular networks, for instance, the objective function is non-differentiable; therefore it is not feasible to implement gradient-based techniques.

To tackle these problems, a number of objectives need to be formulated for an effective evaluation of the use of machine learning techniques in graph data for the detection of anomalous behaviors in dynamic networks.

1.4 Research Objectives

The aim of this research is to investigate different deep learning algorithms for the discovery of anomalous behavior in network activity based on graph data. To reach its goal, we formulate the following objectives:

- To build an anomaly classifier based on deep neural networks for the discovery of malicious behaviors in the network
- To understand under which conditions the algorithm performs better and compare its efficiency against a well-known dataset
- To assess the performance of the anomaly classifier using evaluation metrics associated with standard classification problems

1.5 Research Significance

The main goal in anomaly detection is to answer the question: "What is interesting about a dataset?". The task of anomaly detection is to discover patterns that do not conform to the predefined scope of the data. These patterns are called anomalies or outliers (Agrawal and Agrawal, 2015). Thus, identifying such anomalies can produce a valuable insight into the underlying behavior of the data.

Although various approaches have been proposed for anomaly detection, little research has been conducted on data represented as graphs. With the increasing popularity of graph networks and their vital structural relationships, the need for mining such valuable datasets becomes more evident. There are many benefits of exploring graph theory for anomaly detection applications. First, data units are usually related to one another and have some inter-dependencies, which helps in finding anomalies in interconnected networks. Furthermore, anomaly detection problems generally aim to address relational phenomena. In the

cybersecurity domain, for example, botnets are typically related to each other. Thus, these problems can be easily modeled as graphs to reveal anomalies. Also, graphs have a powerful representation of networks measuring correlations (edges) between related data objects (nodes or vertices).

However, the first step in analysing any data is to find a suitable mathematical representation to handle its underlying structure. In social networks, such data are large (scale of billion), complex and are natural candidates for machine learning methods. Machine learning and deep neural networks have shown a considerable achievement in a broad range of problems and could possibly be generalized to non-Euclidean data. Recent advances in representation learning, such as network embedding and deep neural networks, makes it possible to capture the trends from interactive networks like friendship or collaboration.

In this work, our contribution is twofold. First, we employ the sliding window model to the dynamic network and represent it a sequence of overlapping snapshot graphs to capture the trends between windows. Then, we train a deep neural network on a set of extracted graph-based and time-based features to identify anomalous traffic, and we demonstrate that combining the features improves on the detection performance. The second contribution is, we train a deep neural network on a time-dependent node embeddings using temporal random walks and node2vec and compare the performance with the former approach. The novelty of our work is that we combine graph-based and time-based features for identifying malicious nodes in network graphs using deep learning models. In contrast, researchers mainly focus on either graph-based features or flow-based features. Furthermore, we compare the performance of conventional methods that manually extract the features against the recent representation learning approaches.

1.6 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we present a overview on the state-of-the-art methods for anomaly detection in graph-based data using machine learning. Chapter 3 presents the research methodology along with the rationale for the selection of each component. In Chapter 4, we evaluate the proposed model by illustrating the findings from the experiments and assessing the performance scores through a variety of metrics and referring to related work where necessary. Finally, Chapter 5 concludes with a summary of the research findings and their limitations, as well as providing recommendations for future research.

Chapter 2

Literature Review

Anomaly detection has a big variety of real-life applications, including the early detection of cyber-intrusions and Internet fraud. Although a great deal of research has been done in this area, little of it has focused on data represented as graphs. In fact, graphs can effectively capture correlations and interactions between individual units. Moreover, they provide a powerful reconstruction of networks by varying the visual representation of their nodes and edges. Although they are typically used to visualise networks for human analysis, machine learning can improve on this by exposing knowledge hidden within their structure. Recent advances in research leveraged ML models for anomaly detection over graphs. This chapter provides a comprehensive description of the state-of-the-art methods for anomaly detection in graph-based data under two settings: static graphs and dynamic graphs. It also reviews different approaches of representation learning problems and the associated key difficulties along with available enhancements.

2.1 Anomaly Detection in Static Network Graphs

Many researchers have studied anomaly detection in static graphs. The aim is to find abnormal patterns or unusual events in network units (e.g., nodes, edges, substructures) given the snapshot of a graph. Anomaly detection approaches in static graphs can be categorized as structure-based or community-based.

2.1.1 Structure-based Anomaly Detection

Since graphs can be either plain or attributed, structure-based techniques can be extended to examine deviations in static plain graph and deviations in static attributed graphs.

For anomaly detection methods in static plain graphs, many researchers extract different graph-related features (e.g., node degree, centrality, egonet) to construct a feature space. These graph-centric features have been used in many information security studies, such as spam detection (Becchetti et al., 2006) and network-based intrusion detection systems (Ding et al., 2012). Hassanzadeh et al. (2012) introduced an anomaly detection method based on graph-theoretic features such as betweenness centrality, community cohesiveness, and the number of adjacent nodes and vertices to detect anomalous nodes in social networks. They used graph features related to a single node and its egonet to detect common structures. It was found that although anomalous users hide their identity by providing false information, graph-based approaches can reveal meaningful patterns in the links they establish with other users.

Other anomaly detection techniques take advantage of the graph structure to determine the closeness between graph units. These proximity-based approaches understand the correlation between graph units such that nearby units tend to belong to the same group (e.g., spam/not spam). One famous graph proximity algorithm based on random walks is PageRank (Brin and Page, 1998). The key innovation in this method is to sample node pairs that tend to co-occur on short random walks. Other proximity measures include the Personalized PageRank (PPR) (Haveliwala, 2003) and SimRank (Jeh and Widom, 2002) and its variants that were introduced by Antonellis et al. (2008); Zhao et al. (2009); Chen and Giles (2013)

For some applications, graph objects exhibit non-unique attributes. An example of attributes is user interest in social networks or time and amount in financial networks, and so on. When compared to static plain graphs, the main idea of static attributed graphs is to identify rare substructures and irregular subgraphs, connectivity-wise and attribute-wise. This can be done by identifying structures that occur infrequently.

Noble and Cook (2003) conducted one of the earliest works in this field and proposed a method for measuring the regularity of a graph using conditional entropy. However, datasets typically have both numerical and categorical attributes and considering edge features as distinct discrete values results in substructures that are all equally anomalous. Davis et al. (2011) addresses this problem by discretizing the numerical attributes.

With the growing research interests on machine learning, many methods have been proposed to detect anomalies on attributed graphs (Ding et al., 2019; Li et al., 2019). These methods generally achieve improve on plain graph approaches because they consider graph structure as well as node information. Researchers have proposed semi-supervised techniques to classify unlabeled nodes using a relatively few number labeled nodes (Kipf and Welling,

2016; Yang et al., 2016; Wu et al., 2019; Rong et al., 2019). One issue with such techniques is that they deteriorate when the training set has class imbalance, which is quite normal in anomaly detection problems due to their rarity.

All these structure-based anomaly detection algorithms demonstrated in this section are for use on static graphs and could be useful for forensic analysis. Also, they could be generalized to various types of data and applications, however, false positive rates could become intolerable.

2.1.2 Community-based Anomaly Detection

In community-based anomaly detection, the goal is to analyze clusters of densely connected nodes to detect the vertex or edge that has inter-cluster connectivity. More specifically, nodes or edges that belong to more than one particular community. In this approach, networks represented as bipartite graphs reveal interesting phenomena to spot anomalies (e.g., publication networks and P2P networks).

Nevertheless, there are two problems that need to be addressed in community-based approaches (Sun et al., 2005). First, it is necessary to identify the neighbourhood of a given node. It is equally important to measure the extent of the node's likelihood to be a bridge node. Matrix factorization is one method to address several clustering problems (Kuang et al., 2012; Wang et al., 2010). The factorization can be defined as $\mathbf{A} = \mathbf{X} \times \mathbf{Y} + \mathbf{R}$, such that $\mathbf{X}$ and $\mathbf{Y}$ are considered the low rank factors and $\mathbf{R}$ is the residual matrix. On the other hand, non-negative matrix factorization (NNMF) enforces non-negativity constraints $\mathbf{X}$ and $\mathbf{Y}$, which is effective in identifying communities. Adding non-negativity constraints on $\mathbf{R}$ helps in spotting anomalies, such as DDoS and port scanning activities. In cybersecurity applications, the "bridge" nodes could represent intrusive connections (Ding et al., 2012). Studies show that community-based anomaly detection methods in static plain graphs prove to be effective for identifying malicious or unwanted traffic (Akoglu et al., 2015).

Graph anomaly detection have been studied extensively on plain graphs, but recently, algorithms have been extended to attributed graphs as often observed in real-world scenarios. For attributed graphs, the goal is to detect vertices whose attributes have major differences with respect to other nodes in the same community. Initially, it is important to perform graph clustering based on the structure and then building upon it for the outlier scoring. With *GOutRan*, Müller et al. (2013) proposed an outlier ranking method in subspaces of attributed graphs where nodes are ranked according to their outlierness regarding both graph and attribute properties. Later, Perozzi et al. (2014a) proposed *FocusCO* to detect user-

driven community outliers in attributed graphs. These approaches infer user preference by the "focus" attributes through a set of user-provided exemplar nodes to extract focused clusters and detect outliers.

Recently, residual analysis has been investigated for the problem of anomaly detection in attributed networks. Specifically, Li et al. (2017) proposes *Radar* to match the residuals of attributes with network information for anomaly detection. *ANOMALOUS* (Peng et al., 2018) includes CUR matrix approximation with residual analysis to mitigate noise in the detection of anomalies on attributed networks.

Despite their success, community-based anomaly detection methods on attributed graphs are relatively low due to their shallow mechanisms of handling graph sparsity and data non-linearity.

2.2 Anomaly Detection in Dynamic Network Graphs

Most anomaly detection techniques are based on static graphs that only represent a single snapshot of the given data. However, real-life networks are continually evolving, which demands a shift in interest to dynamic graphs that are capable of evolving over time. Unlike static graphs, dynamic graphs are subject to structural changes when nodes or edges are added and deleted. Possible changes also include modifications of attributes associated with edges or nodes, such as cost or color. There are different types of anomalies in dynamic graphs as objects are added at every point in time. In this section, we categorize anomalies that arise in dynamic graphs into three types: anomalous nodes, anomalous edges, and anomalous subgraphs.

2.2.1 Anomalous node detection

The aim of node-based anomaly detection is to narrow the node set to a subset that contains nodes with irregular behavior only. It is also possible to determine the specific points in time in which nodes are found to be anomalous. Although identifying irregular evolution relies on the approach, it is safe to assume that all methods provide a specific thresholding function for each node. For instance, a function that measures the change of node degree from one point in time to another. A basic definition for a set of anomalous nodes can be described as:

Definition 2.2.1. *(Anomalous nodes).* Given a graph $\mathcal{G}$, a set of nodes $\mathcal{U} = \bigcup_{t=1}^{T} \mathcal{U}_t$, and a specified thresholding function f, the set of anomalous nodes $\mathcal{U}'$ can be measured by

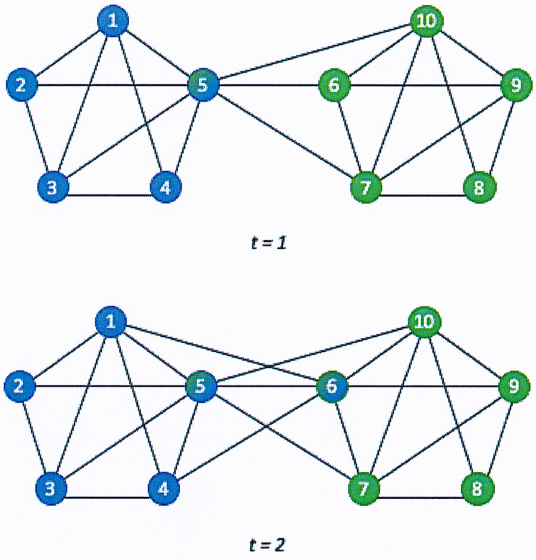


Figure 2.1: An example of anomalous node detection based on community changes

$\left|f(u') - \hat{f}\right| > c_0$, such that $\hat{f}$ is the average of the scores $f(u)$ and c_0 is the threshold for normal versus anomalous data (Ranshous et al., 2015).

As illustrated in Figure 2.1 node u_6 is considered anomalous because it no longer belongs to a single community. In this illustration, an anomalous node has a significant change in its community involvement. When $t = 1$, the node u_6 belongs to community 2 only, but when $t = 2$ the node belongs to both communities 1 and 2. Since u_6's community involvement has changed, it is considered anomalous.

This shift in community involvement is formulated as $f(u) = \sum_{i=1}^{|C|} c_{i,u}^{t} \oplus c_{i,u}^{t-1}$ between successive time points, where C is the set of communities, and $c_{i,u}^{t} = 1$ if v belongs to community c_i at time t and 0 otherwise (Ranshous et al., 2015). Based on the problem at hand, this function f varies. Therefore, if the goal was to identify infected endpoints on a network, then the function f would measure the change in edge weights or the number of edges each node has between time points.

This type of anomaly detection is generally applied for the identification of nodes that are most likely associated with a discovered event like the change in communities in social networks (Akoglu and Faloutsos, 2010; Gao et al., 2010; Ji et al., 2013).

2.2.2 Anomalous edge detection

The goal of anomalous edge detection is to identify edges that have abnormal evolution at a specific point in time. In static graphs, each edge is assigned a value based on the probability of its weight. In dynamic graphs, however, their temporal nature introduces new variants of abnormal edge evolution. First, the edge weight changes over time and might have sudden increase or decrease in value (Li et al., 2009). In addition to that, an edge might appear between two nodes that are normally not adjacent to each other (Abello et al., 2010). In the following is a definition of anomalous edge detection:

Definition 2.2.2. *(Anomalous edges).* Given a graph $\mathcal{G}$, an edge set $\mathcal{E} = \bigcup_{t=1}^{T} \mathcal{E}_t$, and a thresholding function f, the set of anomalous edges $\mathcal{E}'$ is $\left| f(e') - \hat{f} \right| > c_0$, where $\hat{f}$ is the average of the scores $f(e)$ and c_0 is the threshold for normal versus anomalous data.

Figure 2.2 demonstrates the impact of abnormal edge weight evolution in detecting anomalous edges. Typically, the weight of an edge shifts by ± 0.1 between two successive points in time. Yet, at time $t = 2$, the edge connecting v_3 and v_4 has a spike in its weight. Likewise, when $t = 3$, the edge connecting the nodes v_1 and v_4 is flagged as anomalous because it has an abnormal large change in its weight compared to its value at $t = 2$ and $t = 4$.

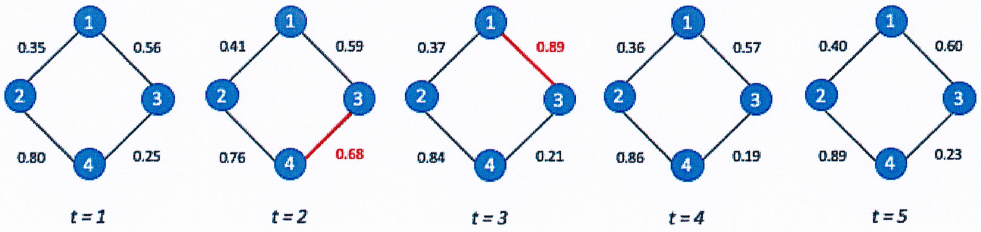


Figure 2.2: An example of anomalous edges based on abnormal weight evolution

One example for using edge-based anomaly detection is tracking the pairwise links of all nodes in the graph by measuring the frequency of times they communicate. Each edge and at every timestep is then scored based on the probability of it being active (Heard et al., 2010). Another example is detecting outliers in vehicle traffic data. In such network, roads could be represented as edges and the intersections could be represented as nodes. The pairwise

similarity between every edge is then calculated by mapping the average speed of traffic during a given time duration (Li et al., 2009).

2.2.3 Anomalous subgraph detection

Although the detection of anomalous edges and nodes have similar concepts, identifying subgraphs with irregular evolution is not as straightforward.

Definition 2.2.3. *(Anomalous subgraphs).* Given a graph $\mathcal{G}$, a subgraph set $\mathcal{S} = \bigcup_{t=1}^{T} \mathcal{S}_t$, where $\mathcal{S}_t$ is a subset of the graph $\mathcal{G}_t$ and a thresholding function f, the set of anomalous subgraphs $\mathcal{S}'$ is ,$\left|f(s') - \hat{f}\right| > c_0$, where $\hat{f}$ is the average of the scores $f(s)$, $\forall s \in \mathcal{S}$ and c_0 is the threshold for normal versus anomalous data.

Figure 2.3 shows two different examples of anomalous subgraphs. When a significant number of nodes are split from a graph as it evolves over time, it is called a *shrunken community*. It is possible to detect such trends by quantifying the decrease in the number of nodes with respect to time evolution. Figure 2.3 also shows an illustration of a shrunken community in which it divides into several smaller communities.

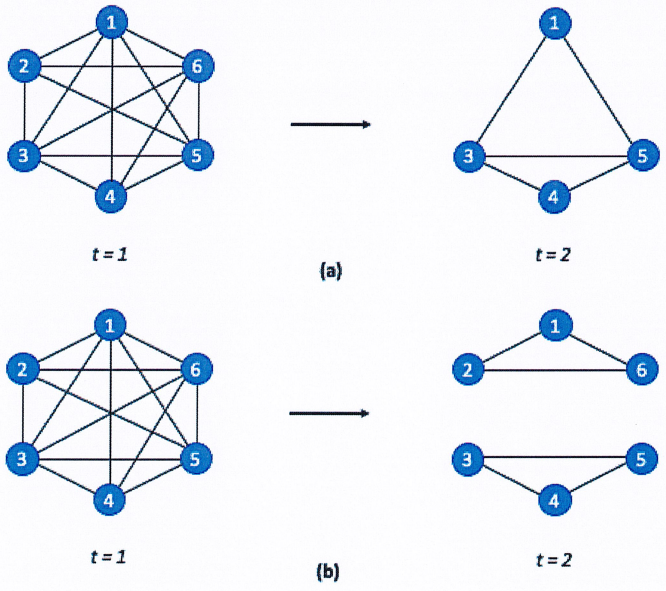


Figure 2.3: Two examples of anomalous subgraphs

There has been extensive work on using subgraphs for spotting anomalies. For instance, Maruhashi et al. (2011) and Pei et al. (2005) study the possibility of identifying novel subgraphs in extremely large networks. Pandit et al. (2007) employs the belief propagation

approach to detect near bipartite cores and associates with fraud, and Beutel et al. (2013) detects near-bipartite cores using edge constraints to discriminate malicious activities. Jiang et al. (2016) proposes CATCHSYNC that exploits two signs of suspicious graph patterns, synchronicity and abnormality, to automatically spot suspicious nodes on large directed graphs.

2.3 Deep Learning on Graphs for Anomaly Detection

A large number of machine learning tasks seek to predict and identify new patterns using data modeled as graphs. Graphs can effectively allow storage and access of relational information between interacting entities (Angles and Gutierrez, 2008). However, the main challenge is to find a method to represent this high-dimensional data about graph structure to incorporate it into a machine learning model.

Conventional approaches mainly rely on user defined features such as degree statistics (Bhagat et al., 2011), kernel functions (Vishwanathan et al., 2010), and node neighborhood (Liben-Nowell and Kleinberg, 2007). However, manually extracting these features is a time-consuming and expensive task. Recently, and with the aid of deep learning, it is possible to seamlessly transform graph structure into low-dimensional embeddings without manual feature extraction. The aim of these representation learning approaches is to encode graph structure by mapping each node (or subgraph) to a low-dimensional vector that summarizes its graph position, its local neighborhood, and its attributes.

2.3.1 Node Embedding

Most node embedding approaches were originally motivated by classic matrix factorization algorithms that are based on dimensionality reduction techniques (Belkin and Niyogi, 2002; Kruskal, 1964). Early methods include Laplacian Eigenmaps, which mainly computes pairwise similarity to determine the quality of the reconstruction. Recent methods, on the other hand, use inner-product to get a reconstruction of the similarity between two vertices in the original graph. In particular, a strong relationship is derived from the dot product of the corresponding embeddings. Methodologies based on an inner-product decoder include the Graph Factorization (GF) algorithms (Ahmed et al., 2013), GraRep (Cao et al., 2015), and HOPE (Ou et al., 2016). These algorithms optimize for the mean-squared-error (MSE) over pairwise connections in the neighborhood to determine the quality of the reconstruction. They are mainly different in the way of measuring the pairwise-similarity of graph vertices. The Graph Factorization algorithm directly uses the adjacency matrix, $\mathbf{A}$, as a similarity

measure. On the contrary, to capture higher-order node similarity, GraRep employs multiple powers of the adjacency matrix, (e.g., $\mathbf{A}^2$); and the HOPE algorithm uses Jaccard neighborhood overlaps. The aim of these algorithms is to generate embeddings for each node in order to approximate node similarity using the inner product between those generated embedding vectors.

Recently, a great deal of promising algorithms have discussed the potential of learning node embeddings through random walks, such as node2vec (Grover and Leskovec, 2016), LINE (Tang et al., 2015), and DeepWalk (Perozzi et al., 2014b). Unlike matrix factorization-based algorithms that use a deterministic calculation of node similarity, these random walk approaches enhance the performance by employing a flexible and stochastic measure (Goyal and Ferrara, 2018). In particular, their key innovation is to sample pairwise connections that tend to happen on short random walks, and then optimize for their similarity in the embedded space. Random walk approaches are inspired by the skip-gram model from Natural Language Processing (NLP) (Mikolov et al., 2013b).

Chen et al. (2018) have proposed HARP, a hierarchical representation learning process to improve the performance of node2vec, LINE, and DeepWalk by collapsing similar vertices in the graph together into "supernodes" and then using the learned embedding as a first value for the random walk on a coarsened version of the graph. There are many other extensions to the random walk methods, including but not limited to the works proposed by Perozzi et al. (2016) and Chamberlain et al. (2017)

2.3.2 Inductive Representation Learning

One drawback of the aforementioned methods is that they learn static embeddings on graph snapshots; hence, they are inherently transductive (Hamilton et al., 2017a). In other words, they only assign embedding values for vertices that were active in the training. This is a drawback that is extremely important for dynamic and evolving graphs that cannot be fully stored in memory.

Nevertheless, some extensions were proposed to facilitate online learning from graph streams. An online incremental update has been proposed but not analyzed for DeepWalk (Perozzi et al., 2014b). Yu et al. (2018b) proposed an incremental model (DLNE) with a batch of edge additions and deletions based on LINE, but the online, single edge insertion behavior was not analyzed.

On the other hand, a number of recent embedding methods have been presented to address the problem of generalizing to new graphs after training. These techniques are based on deep

neural networks and typically take advantage of graph structure and attributes. It is still hard to use deep node embedding methods for time-evolving graphs. As dynamic graphs evolve over time, it is important to represent the temporal changes in the embedding space.

2.3.3 Anomaly Detection over Graph Embedding

Some recent works have investigated incorporating node embeddings into anomaly detection. Sricharan and Das (2014) uses commute time distance to identify abnormal differences in dynamic graphs. However, the method focuses on structural differences and cannot detect long-term anomalies. DynGEM (Goyal et al., 2018) applies node embeddings and classifies the difference in embeddings between consecutive time points as anomaly. Mahdavi et al. (2018) proposes dynnode2vec, a dynamic embedding method based on node2vec and initializes the current graph embedding with previous embedding vectors. Yu et al. (2018a) proposed NetWalk, a dynamic network embedding technique for anomaly detection based on deep neural networks and random walks. The anomaly detection is realized using a clique embedding technique along with reservoir sampling that updates the representations over time, and finally uses a dynamic clustering model. AddGraph (Zheng et al., 2019) extends the original GCN model to support temporal information using attention-based GRU (Gated Recurrent Unit) and introduces a selective negative sampling strategy and margin loss in model training for detecting anomalous edges.

Generally, there is a lot of work that has been done on anomaly detection and there is a possibility to leverage graph-based data for training neural networks. In our work and based on the problem at hand, the goal is to detect anomalous nodes and be able to distinguish benign hosts from malicious hosts in a given network. Therefore, we analyze the performance of a deep neural network trained on a set of graph-based and time-related features.

Chapter 3

Methodology

In this chapter, we describe the anomaly detection approaches that were applied on the CIC-IDS2018 dataset (Sharafaldin et al., 2018). The first approach leverages graph-based and other time-related features to detect anomalous activities in the network. On the other hand, in the second approach, we propose an anomaly detection method that learns time-dependent node embeddings.

3.1 Feature-based Anomaly Detection

In brief, the proposed approach examines network graphs over time to determine which nodes are most likely to be malicious by generating graphs at evenly distributed intervals and extracting useful graph-based and time-related flow-based features. Then, a time series is assembled based on the extracted features for each host. Finally, we train a deep classification model to detect malicious nodes. Figure 3.1 illustrates the basic structure of the proposed anomaly detection classifier.

Since a graph-based approach is employed, the detection approach does not rely on the packet content but rather targets the communication structure. Therefore, this will reduce the likelihood of evading detection through packet encryption. In addition, our approach could be generalized and extended to many different attack scenarios since communication structures are typically very similar in botnets.

3.1.1 Motivation

In this section, we describe the motivation of the system with an example derived from the packet capture file during the time $200s$ to $1250s$. Figure 3.2 presents the evolution of the

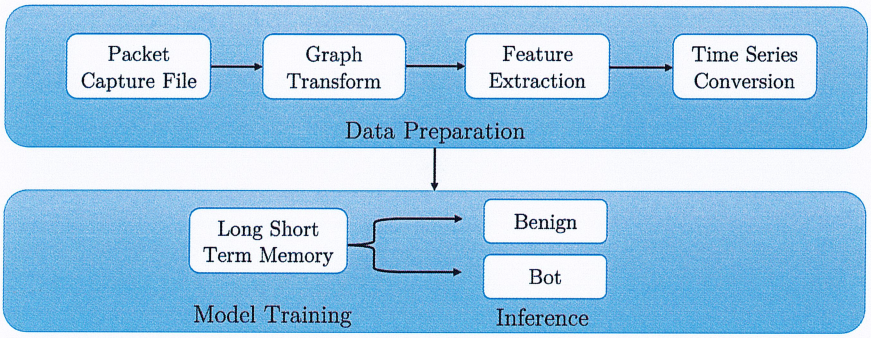


Figure 3.1: Components of the anomaly detection system

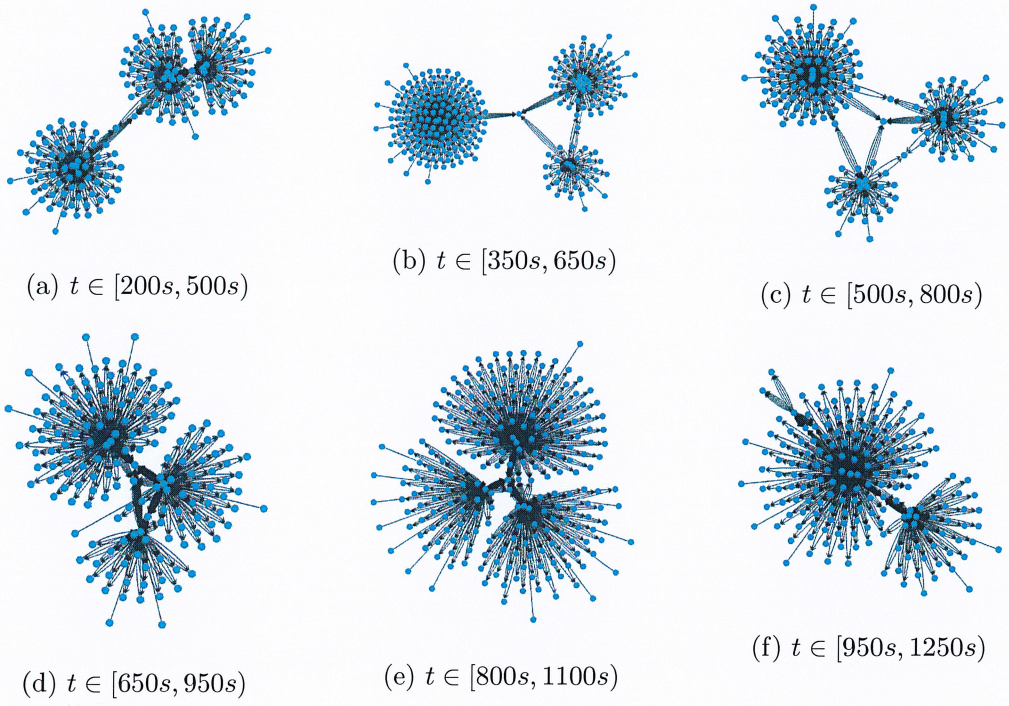
network graph over time. Each snapshot represents the transactions over a $300s$ interval. Botnets typically have a periodic cycles of activity and inactivity. Therefore, the intuition is that a graph-based anomaly classifier is powerful for revealing malicious behavior.

3.1.2 Graph Transform

In this study, we employ the sliding window approach to model the dynamic network. In essence, the input packet capture file is split into overlapping time intervals (i.e., *windows*); each window is x seconds long and overlaps with $\frac{X}{2}$ seconds of the previous window. The size of the window is the duration of a subset of the file that is used to create a single graph. In order to transform the raw packet capture into a network graph, we model the time-evolving network as a sequence of individual snapshot graphs $< G_1, G_2, ... >$ and the active window G_i is $< g_{t-L+1}, g_{t-L+2}, ..., g_t >$ where L is the length of this window and t is the duration of the packet capture file.

The input to the system are the raw data packets that are transformed to edges as 6-tuples list of the form: `(src_ip, dst_ip, timestamp, src_port, dst_port, ttl, num_bytes)`, where `src_ip` is the source IP address of a source host, `num_bytes` is the number of data packets exchanged by `src_ip` and `dst_ip`, the destination host IP address. Similarly, `src_port` is the port number of the source host and `dst_port` is the destination's port number. The `ttl` is the time-to-live, which determines the amount of time that a packet is expected to exist inside a network. `timestamp` is basically the number of seconds that have elapsed since Unix epoch.

We then use this graph to extract features for each node present during this particular



(a) $t \in [200s, 500s)$ (b) $t \in [350s, 650s)$ (c) $t \in [500s, 800s)$

(d) $t \in [650s, 950s)$ (e) $t \in [800s, 1100s)$ (f) $t \in [950s, 1250s)$

Figure 3.2: A sequence of graph snapshots generated from the packet capture file of the botnet category

window. The window size is selected such that it is large enough to realize trends in a single graph and small enough that trends are not meaningless. In addition to the window size, the step size is another hyper-parameter that controls the time-series construction. The step size can be defined as the duration of overlaps between windows. This parameter enables gradual transition such that potential trends between windows do not go unnoticed.

The implementation was done using Graph-tool (Peixoto, 2014), which is an efficient Python library (based on C++) for network analysis and visualization, to generate graphs from raw packet capture files as well as extracting the required graph-based features.

3.1.3 Feature Extraction

In this step, we start extracting the features that we consider in our work. The aim is to create the required graph-based features for training the neural networks. Features are a success factor for ML models that are able to represent and discern host behavior, especially that of bots. However, it is vital to be efficient when extracting graph-based features as improper selection of features may result in increased computational cost. The seven graph-

based features used in this study are in-degree, out-degree, betweenness centrality, eigenvector centrality, PageRank, authority and hub centrality. In the following, we briefly discuss the selected graph-related features and the rationale of their selection, as well as the proposed flow-based features that are based on time.

In-Degree

When many suspected infected machines communicate with a malicious C&C server, this most likely results in a relatively high in-degree value for the botmaster. Thus, in-degree has been selected as a feature to identify botnet activity in a network. For a particular host in a directed graph, in-degree represents the total number of head ends adjacent to that node (see Definition 1.2.1). High in-degree values indicate that the node tends to establish more connections, which is a typical botnet behavior. Also, in-degree is specifically important for detecting DoS and DDoS attacks. This is due to the fact that in such scenarios, the target is flooded with bogus traffic, and hence, has a very huge in-degree value.

Out-Degree

The out-degree is the total number of tail ends adjacent to a node. Similar to the in-degree, a high out-degree value also indicates the that the node tends to establish more links with other nodes. Bots usually initiate connections with other infected machines to expand the reach of the botnet network. Therefore, out-degree is leveraged as a helpful indicator of botnet activity in a graph.

Betweenness Centrality

In graph theory, the betweenness of an edge is the number of shortest paths that traverse it (Tyler et al., 2003). In other words, it is how many times a node becomes a bridge along the shortest path between two other nodes. It is a measure of the number of shortest paths that crosses a particular node. Betweenness centrality is useful in identifying P2P botnets because they are interconnected with no centralized botmaster. Therefore, we predict that a P2P bot is more likely to have a big betweenness centrality value in a graph.

PageRank

Another centrality-based score function that has been designed to calculate the importance of a node in a graph is PageRank (Brin and Page, 1998). Here, we leverage PageRank as a

node scoring function because it is fast to compute and it is intuitive, a node is important if its neighbours are important. Needless to say, it led to the very successful ranking of Google's search engine.

Eigenvector Centrality

Eigenvector centrality, also inspired from social network analysis, measures the influence of a node in a network. In essence, it is the relative weight of a vertex, such that links to high-scoring nodes have greater contribution to its score (Langville and Meyer, 2005). We expect that the eigenvector centrality of an infected host to be different than that of a normal host. Therefore, it is selected as a feature to detect malicious activities in this study. During the implementation, it was noticed that the eigenvector algorithm in graph-tool has a very slow convergence, hence, a fixed value of maximum iteration was set to 500 to overcome the issue.

Authority Centrality and Hub Centrality

In a directed network, the in-degree of node v alone does not necessarily reveal any insights of the relevant nodes to it. Similarly, the out-degree does not necessarily unveil the important nodes to node v. Authority is how much information held by a node on a topic. On the other hand, hub centrality measures how well a node 'knows' where to find information on a given topic. Kleinberg et al. (1998) proposed an iterative process to assist in hyperlinked web pages queries and determine which pages are authoritative and which pages are hubs to authoritative pages. The same concept can be leveraged for examining botnet communications.

Time-based features

To improve the accuracy, we employ additional flow-based features that are based on *time*. As illustrated in Table 3.1, we leverage time-related features to identify anomalous traffic. In some features we measure the time, e.g. time that a packet is set to exist inside a network before it is discarded (i.e., TTL).

The rationale behind this selection is that if we identify various TTL values for a given session, it could indicate that the route has changed and packets no longer follow the same path as they did at the beginning of the connection. Although this could legitimate changes, it could also mean an attacker changed the route the packets take (e.g., a Man-in-the-Middle attack). Moreover, specially crafted TTL values are associated with TTL expiry attacks, one type of DoS attacks (Vlajic et al., 2012). When a packet expires due to its TTL reaching

Table 3.1: Sample of extracted time-related features in the CIC-IDS2018 dataset

Feature Name	Description
avg_incoming_packet_size	The average incoming packet size of node v_i per interval
avg_outgoing_packet_size	The average outgoing packet size of node v_i per interval
number_incoming_bytes	The number of incoming bytes to node v_i per interval
number_outgoing_bytes	The number of outgoing bytes from node v_i per interval
avg_incoming_ttl	The average incoming time-to-live per interval
avg_outgoing_ttl	The average outgoing time-to-live per interval

the value zero, an "ICMP TTL Exceeded" message is sent to the sender. This mechanism could be exploited by attackers to perform DoS attacks by sending a flood of traffic with the TTL value set in a way that it expires on the target. Therefore, the target has to reply with a large number of "ICMP TTL Exceeded messages", which consumes the target's resources. Figure 3.3 shows an illustration of a TTL expiry attack.

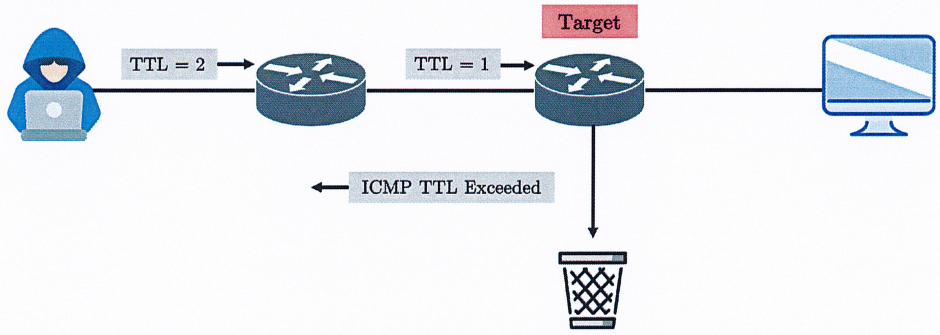


Figure 3.3: An example of a TTL expiry attack

Other time-related features were extracted in which we fix the time (i.e., interval) and measure other variables, e.g., bytes per interval or packets per interval. Instead of analyzing features of the network traffic as a whole on each host, we analyze time-related features on a per-host basis (see Table 3.1). After splitting the network activity into time windows of length t, within each window, we measure the total number of bytes and packets transferred and the ports used by each host. Figure 3.4 shows the overall traffic in the packet capture file measured represented in bytes per minute during different timestamps. This is important because P2P applications will normally have large data sizes as these applications consume

a large amount of bandwidth (e.g., streaming). On the other hand, attackers use small transmissions to minimize their footprint in the environment and maintain the connection unnoticeable. (Almutairi et al., 2020). Furthermore, small packet sizes improve the efficiency of the connection between bots and the C&C server.

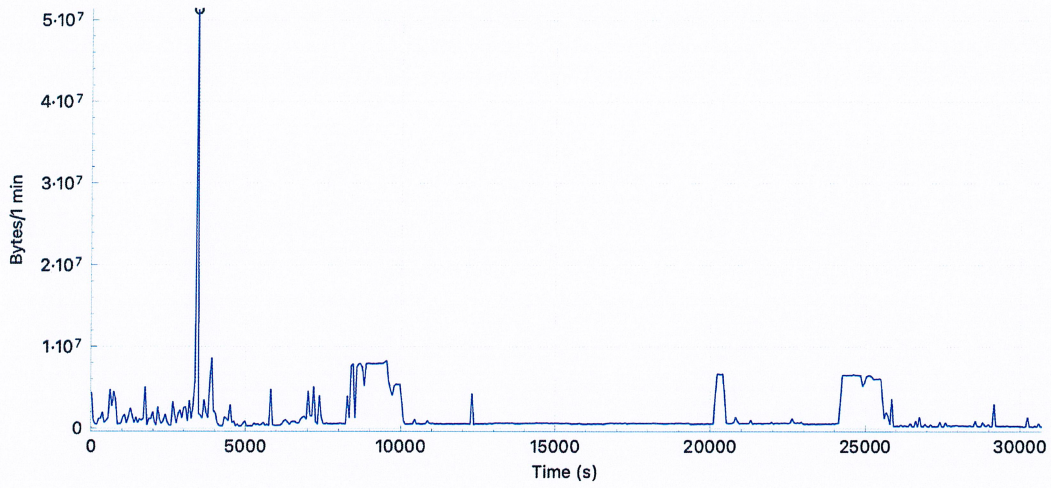


Figure 3.4: The overall traffic in the capture file measured in bytes per minute

3.1.4 Feature Normalization

The extracted features are the input to the neural network, and the output is the class label indicating the nature of the flow (i.e., malicious or benign). The features are first normalized from 0.05 to 0.95 using the following equation since. During the implementation, the *sigmoid* function uses the normalized features to map the weighted inputs to the output layer.

$$f_n = 0.05 + 0.9 * \frac{f - f_{min}}{f_{max} - f_{min}} \tag{3.1}$$

3.1.5 Time Series Conversion

A time series is a sequence of observations listed at successive equally spaced points in time (in a chronological order). We compute the time series for each unique host during each interval. However, some hosts are not active in every interval, therefore, we zero-pad them in whenever they do not send or receive packets. Given that the number of generated graphs from a packet capture file N_G is equal to $\frac{t}{s} - 1$, where t is the duration of the packet capture file and s is the selected step size, the following equation represents the time series assembly

of the network graphs for node v_i. The features are denoted as f_{x,N_G}, where x represents the number of features and N_G represents the number of generated graphs.

$$v_i = [f_{1,1}, f_{2,1}, ..., f_{x,1}], [f_{1,2}, f_{2,2}, ..., f_{x,2}], ..., [f_{1,N_G}, f_{2,N_G}, ..., f_{x,N_G}] \tag{3.2}$$

3.1.6 Long Short-Term Memory Model

LSTMs are a special type of Recurrent Neural Networks (RNN) proposed by Hochreiter and Schmidhuber (1997). The reason LSTM is selected to create our models is because it is a good candidate for processing time-correlated objects such as network traffic (Lin et al., 2019). In this work, we follow the supervised learning approach to build classifications models and consider the anomaly detection task as a binary classification task. This approach relies on labeled data using ground truth information published by the creators of the dataset. The implementation of the LSTM in the experiments is based on Keras, a deep learning (Application Programming Interface) API written in Python (Chollet et al., 2015).

The main problem of leveraging LSTM for building anomaly detection models is caused by the size and proportion of the historical data used. A known limitation of supervised learning methods is their need of balanced labeled data. In particular, it is necessary that the number of malicious hosts is relatively equivalent to the number of the benign hosts. Otherwise, the learned model would normally be biased towards the majority class.

3.2 Anomaly Detection using Node Embeddings

In this section, we provide an overview of anomaly detection method that is based on node embeddings. Figure 3.5 explains the components of the embedding-based anomaly detection. After reading the packet capture file and transforming it into a graph, we perform time-respecting walks. We then train a node2vec model to generate embeddings. Finally, a logistic regression classifier is used for class prediction.

3.2.1 Temporal Random Walk

Most researchers model dynamic networks as a separate graph snapshots, similar to our approach in the previous section, which is $\mathcal{G}_1, ..., \mathcal{G}_T$ where $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}_T)$ and $\mathcal{T}$ is the timeframe $[t_{i-1}, t_i]$. However, in this section, we incorporate temporal dependencies into network embedding methods to improve the performance of the anomaly detection models. Specifically, we create a time-dependent network representation that examines the significant temporal

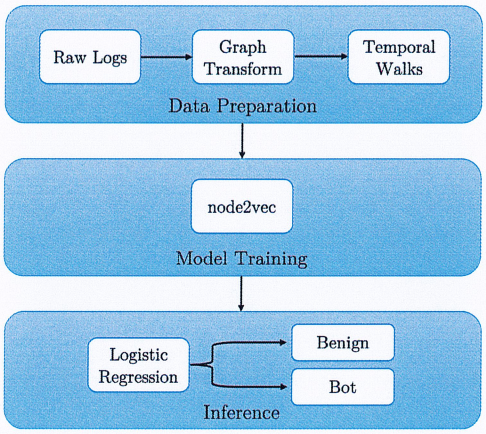


Figure 3.5: Components of the anomaly detection system

attributes of the dynamic network described by Nguyen et al. (2018). This can be defined formally as:

Definition 3.2.1. *(Continuous-Time Dynamic Network).* Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_T, \mathcal{T})$, where the node set $\mathcal{V}$, and $\mathcal{E}_\mathcal{T}$ is the set of temporal edges, every edge $e_i = (u, v, t)$ has a discrete time t.

In temporal networks, a *valid* walk traverses nodes connected by edges whose timestamp is greater than the present edge. This improves on embeddings based on random walks that do not take into consideration the time when examining a sequences of events. A temporal walk by definition adds an extra constraint to the normal random walk by forcing the walk to respect time when traversing nodes.

3.2.2 node2vec

After generating the random walks, we train a node2vec model to generate node embeddings. node2vec (Grover and Leskovec, 2016) is one of the most popular node embedding algorithm. It was introduced as an improvement to DeepWalk, which is another embedding algorithm that is based on random walks. The node2vec method has two main steps, namely: random walk generation and skipgram (Mikolov et al., 2013a). In the first step, node2vec generates random walks for every node in the graph. Then, it explores the node's neighbourhood by

traversing the graph using DFS and BFS. To achieve that, it introduces two parameters p and q, which control the search strategy. In the second step, a skipgram model is trained using the generated walks from phase one. Finally, the model learns the representation for every node.

3.2.3 Logistic Regression

Logistic regression is a useful method for classification problems. Here, we use the binary variant of logistic regression because we are expecting two possible outcomes (i.e., bot or benign).

3.3 Summary

In this chapter, we described the ML-based anomaly detection approaches that were applied on the CIC-IDS2018 dataset (Sharafaldin et al., 2018). The first approach leverages graph-based and other time-related features to detect anomalous activities in the network. On the other hand, in the second approach, we propose an anomaly detection method that learns time-dependent node embeddings.

In the first approach, the input packet capture file is split into overlapping time intervals. We then use each graph to extract features for each node present during the snapshot. Next, we compute the time series for each unique host during each interval. Finally, LSTMs are used to for building the anomaly detection models.

In the other approach, the anomaly detection task is based on node embeddings. After reading the packet capture file and transforming it into a graph, we perform time-respecting random walks. We then train a node2vec model to generate embeddings. Finally, a logistic regression classifier is used for class prediction.

The next chapter presents the results of each approach and discusses the limitations. Also, it compares the obtained results with the literature.

Chapter 4

Results and Discussion

In this chapter, we briefly describe the dataset and the performance metrics used in the experiments. We also discuss the implementation and evaluation of the anomaly detection classifier. Then, we present the experimental setup and the results of our assessment by comparing two graph-based approaches: conventional feature-based anomaly detection and embedding-based anomaly detection.

4.1 Dataset

Anomaly detection has been an interesting research area due to its potential in detecting novel attacks. The behavioral patterns of anomalous activities change over the years, and therefore, it is essential to upgrade the conventional datasets for evaluating the effectiveness of an anomaly-based IDS.

We trained and tested our models on the CIC-IDS2018 dataset. The CIC-IDS2018 dataset (see Table 4.1) was created by The Canadian Institute for Cybersecurity (CIC). It includes seven attack categories: brute-force, heartbleed, botnet, DoS, DDoS, web attacks, and network infiltration. The daily recorded data consists of the raw packet captures and event logs of each machine.

In this work, the scope is limited to two scenarios only, DDoS (Distributed Denial of Service) and botnet attacks since they have a higher number of malicious hosts than the other scenarios. This mitigates the data imbalance issue described in Subsection 3.1.6. Table 4.2 presents the details of the selected attack categories.

To further alleviate the data imbalance issue, we performed undersampling and exempted many benign traffic in a 10:1 ratio as depicted in Figure 4.1.

Table 4.1: Specifications of CIC-IDS2018 Dataset

Dataset Name	CIC-IDS2018
Dataset type	Multi-class
Year	2018
Duration of Capture	10 Days
Attack Infrastructure	50 PCs
Victims	420 PCs and 30 servers

Table 4.2: Details of the selected attack categories

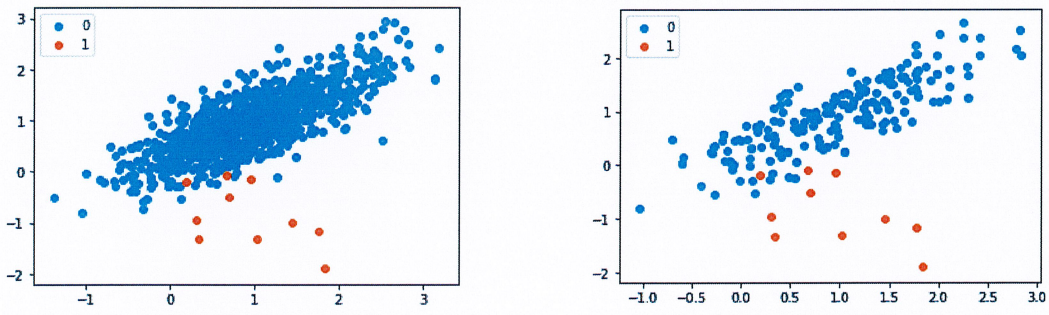
Date	Attack Type	Start Time	Finish Time	# of Bots
21-02-2018	DDoS	10:12	11:10	10
21-02-2018	DDoS	13:13	13:32	10
02-03-2018	Botnet	10:11	11:34	10
02-03-2018	Botnet	14:24	15:55	10

4.2 Performance Metrics

The metrics used to evaluate and compare the performance of the experiments are accuracy, precision, and recall. TP, TN, FP, FN represent True Positive, True Negative, False Positive and False Negative respectively.

Accuracy measures the proportion of correctly classified samples, and is formulated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



(a) Before undersampling

(b) After undersampling

Figure 4.1: Scatter plot of imbalanced dataset before and after undersampling

Precision measures how many actual positives were correctly identified out of all positive samples, and by definition it is:

$$Precision = \frac{TP}{TP + FP}$$

Recall represents the system's ability in detecting anomalies. It can be defined as:

$$Recall = \frac{TP}{TP + FN}$$

The Receiver Operating Characteristics (ROC) curve is also used because accuracy by itself may not be a good evaluation metric when the data is imbalanced. ROC is one of the most prominent evaluation metrics for assessing the performance of classification models.

4.3 Results

In this section, we compare the experimental results of implementing the anomaly detection approaches discussed in Section 3. More specifically, we compare their performance by varying their hyper-parameters to obtain an optimal classifier.

4.3.1 Performance of Feature-based Anomaly Detection

LSTMs in Keras can be built in two ways, *stateful* and *stateless*. Here, we compare the performance of stateful and stateless LSTM networks for anomaly detection. Moreover, we compare the results with varied window sizes and step sizes to demonstrate the influence of these hyper parameters on performance. In both stateless and stateful experiments, the data is split into a training set with the size of 70% while the remainder 30% is assigned to the test set.

A *stateless LSTM* is the default configuration in Keras, where the LSTM cell memory is reset in every batch. Therefore, batch size selection is very crucial in the stateless network. In the experiment, we use a single LSTM layer with 64 hidden units. Then, the output of the model is flattened to a single one-dimensional vector before a prediction is made. We train the model using the ADAM (Adaptive Moment Estimation) optimizer for 200 epochs (Kingma and Ba, 2014) along with the mean squared error loss function.

Table 4.3 represents the performance of the stateless network and provides a summary of the performance results on each dataset using `window_size` = 300 and `step_size` = 150 in the first run. Next, we increase both `window_size` = 500 and `step_size` = 250 to showcase the importance of time selection on performance. It was found that the accuracy is

Table 4.3: Performance of the stateless LSTM on the analyzed attack categories

Metrics	Bonet		DDoS	
	W=300	W=500	W=300	W=500
Accuracy	93.5 %	93.1 %	93.5 %	94.0 %
Precision	0.45	0.43	0.44	0.47
Recall	0.99	1	0.99	1
AUC	0.99	0.98	0.97	0.98

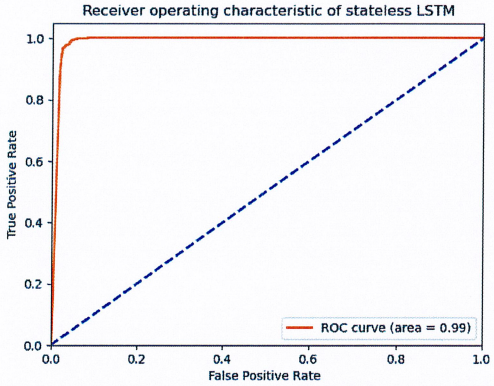
Table 4.4: Performance of the stateful LSTM on the analyzed attack categories

Metrics	Bonet		DDoS	
	W=300	W=500	W=300	W=500
Accuracy	100 %	98.9 %	99.2 %	99.6 %
Precision	1	0.87	1	0.95
Recall	1	1	0.99	1
AUC	1	0.99	1	1

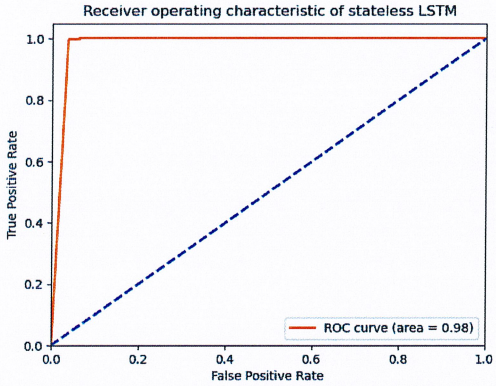
in the range of 93% to 94%, which is relatively low. Moreover, the high recall-low precision negatively impacts the quality of the system and essentially indicates that the classifier returns a lot of false positives. This issue is common in dataset with disproportional class distributions even though undersampling techniques were performed.

When using *stateful LSTM* networks, the output hidden states and cell states of $batch_{i-1}$ are the initial states for $batch_i$. In the experiment, we use a stacked-LSTM network that consists of two LSTM layers with 64 hidden units. To prevent overfitting, a dropout layer is added after each LSTM layer and the value is set to 0.5. The model is also fit through the efficient ADAM optimizer (Kingma and Ba, 2014) and the mean squared error loss function.

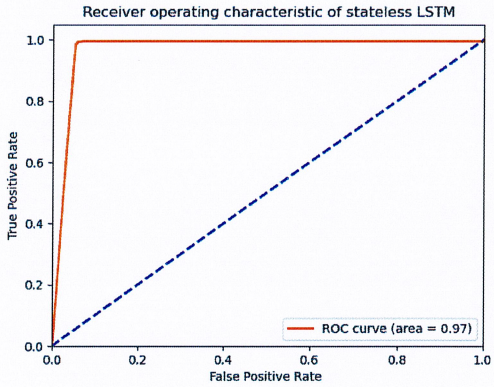
In Table 4.4, a summary of the experimental results is given. The results suggest that the stateful LSTM configurations outperforms the stateless configuration in the problem at hand. In other words, the stateful variant is more suitable for the anomaly detection task. When it comes to accuracy, the best achieved result is from the botnet scenario in which it was 100%. The high recall and high precision values indicate that the classifier has a good performance and correctly identifies almost all malicious hosts as malicious.



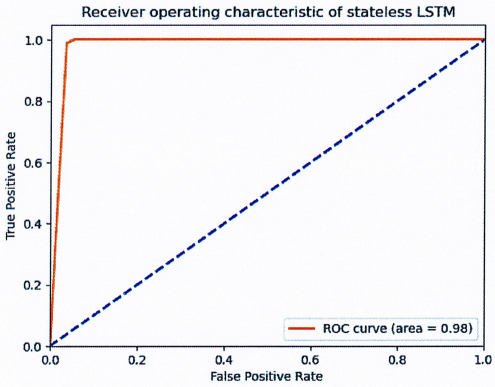
(a) Botnet ($W = 300$ and $S = 150$)



(b) Botnet ($W = 500$ and $S = 250$)



(c) DDoS ($W = 300$ and $S = 150$)

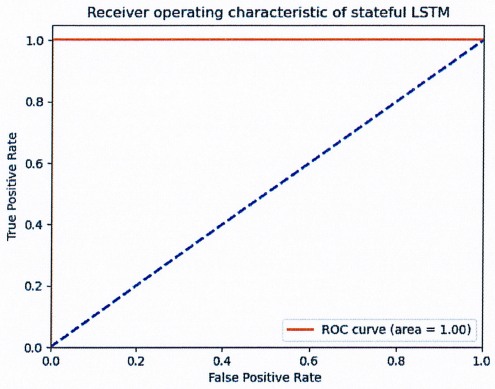


(d) DDoS ($W = 500$ and $S = 250$)

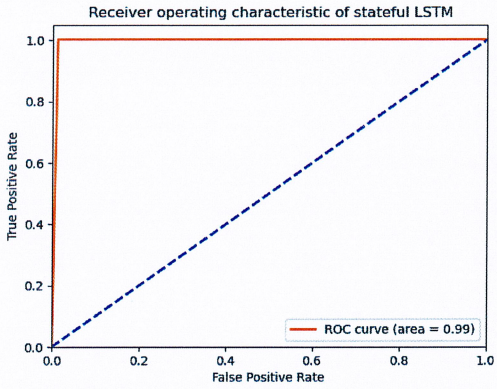
Figure 4.2: ROC curve of the stateful model trained on the analyzed attacks

Receiver Operating Characteristics (ROC) Analysis

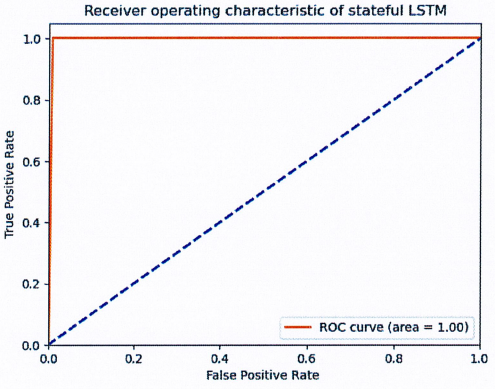
Figure 4.2 shows the true positive rate (sensitivity) versus false positive rate (specificity) of the stateless LSTM model at two different window sizes and two attack categories. The curve implies how well the binary classifier discriminated between the two classes (i.e., malicious or benign). Using this configuration, we can say that our model generalizes relatively well since the area under the ROC curve (AUR) is in the range of 0.97-0.99. However, we can not neglect the fact that this model has a high false positive rate (FPR) which is 9.56% in average.



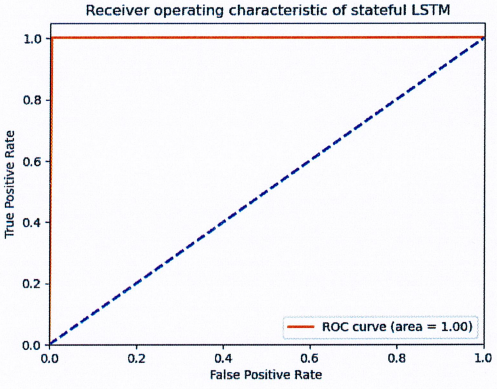
(a) Botnet ($W = 300$ and $S = 150$)



(b) Botnet ($W = 500$ and $S = 250$)



(c) DDoS ($W = 300$ and $S = 150$)



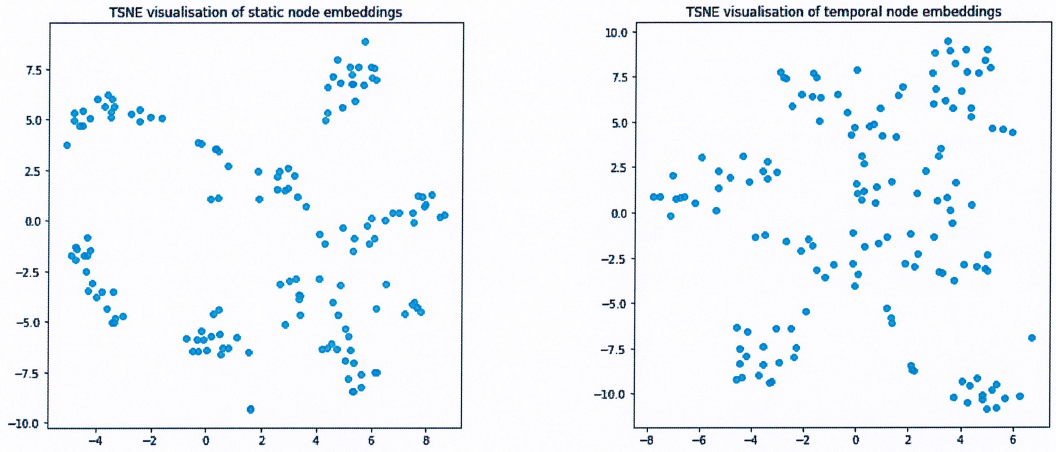
(d) DDoS ($W = 500$ and $S = 250$)

Figure 4.3: ROC curve of the stateful model trained on the analyzed attacks

The stateful variant overcomes this issue and has near-optimal curve (see Figure 4.3). The AUROC in three out four experiments is 1. The average FPR among all scenarios is 2.19%, which is very good compared to the stateless configuration. More importantly, in the botnet scenario (Figure 4.3a), the FPR is almost zero. This proves that our model generalizes strongly. It is worth noting that setting `window_size` = 300 and `step_size` = 150 achieves the best overall results.

4.3.2 Performance of Anomaly Detection using Node Embeddings

The implementation of this experiment is also based on Python. To develop the comprehensive approach for data analysis and visualization and run large scale graph analysis, we use



(a) Static embeddings (b) Temporal embeddings

Figure 4.4: A tSNE visualization of the node embedding for the hosts

StellarGraph (Data61, 2018). It is a library for building machine learning models on graphs. In this experiment, we use StellarGraph to explore how time-dependent random walks that are derived from the packet capture graph can be utilized to generate network embeddings for an anomaly detection task.

In order to perform random walks, we need to specify a number of parameters. One of these is the number of walks per node, which we set it to 10. Moreover, we need to specify the walk length, and in our case is set to 50, because walks will not be allowed when no edges with greater timestamps exist.

Subsequently, we train a node2vec model to create embeddings and set the context window to 10. This parameter defines the number of nodes that will be used to determine the context of each node. Since we are dealing with a binary classification task that does not require a very high-dimensional representation, we set the embedding size to 128. It is also worth noting that the embedding of nodes that are not covered by the walks (e.g., because they are very short) are zeroed.

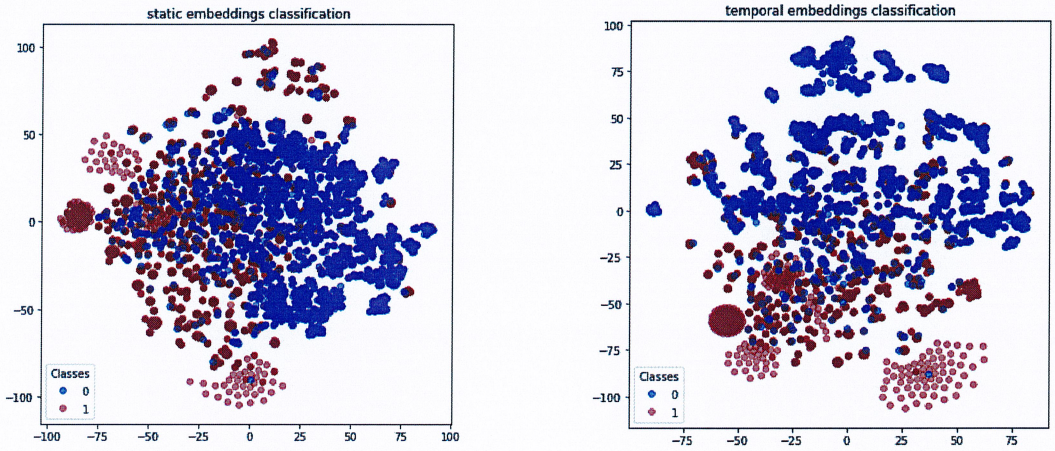
Figure 4.4 displays a visualization of the node embedding for the hosts in the packet capture file. As illustrated, the temporal walks produced a more clustered hosts.

After that, the resulted node embeddings are input to the anomaly classifier. We use the logistic regression with 10-fold cross validation. As displayed in Table 4.5, the AUC scores of the anomaly classifier using temporal embeddings for the botnet and DDoS datasets are: 0.74 and 0.71, respectively. On the contrary, the anomaly classifier using static embeddings achieved 0.69 and 0.68 AUC scores for the botnet and DDoS datasets, respectively.

Table 4.5: Performance of the anomaly classifier based on node embeddings

Metrics	Bonet	DDoS
AUC (temporal embeddings)	0.74	0.71
AUC (static embeddings)	0.69	0.68

As shown in Figure 4.5, it was observed that temporal embeddings have better clustering capabilities for nodes within the same class (i.e., malicious or benign). On the contrary, the visualization of the classification based on static embeddings produces a sparser representation. Of course, oversampling was performed a priori to account for the class imbalance. This justifies the large number of nodes belonging to the minority class (malicious).



(a) Classification based on static embeddings (b) Classification based on temporal embeddings

Figure 4.5: A visualization of the classification based on node embedding

Although very promising, anomaly detection using node embeddings still calls for further exploration and investigations. Therefore, and due to the low AUC scores, we only consider the results from Section 4.3.1 in further comparisons.

4.4 Discussion

In this section, we compare our results to related research that use the same dataset. All scores acquired from the metrics (accuracy, precision, recall, and AUC) are the optimal scores for CIC-IDS2018. Table 4.6 provides a listing by author of similar work along with the best

performance scores for binary classification. In Atefinia and Ahmadi (2021), the researchers integrate four components including two LSTMs and a GRU. The attack categories that resulted in the highest accuracy values are DoS, DDoS, and brute force. One drawback is the lack of score that conveys the average performance of the attacks in total. Catillo et al. (2020) leveraged autoencoders for training the anomaly classifier and found that the botnet attack resulted in the top accuracy. However, the authors used a single classifier for the anomaly detection task. D'hooge et al. (2020) trained 12 different supervised algorithms on CIC-IDS2018 as well as the former release, CIC-IDS2017, to examine how well the classifier can generalize. It was found that the model cannot generalize to other scenarios when a different dataset is used due to the number of unique values in categorical data. Fitni and Ramli (2020) followed an ensemble learning approach using Gradient Boosting, Logistic Regression, and Decision Tree. Kanimozhi and Jacob (2019) implement a two-layer MLP classifier for botnet detection. In this study, a very high performance scores are achieved. Nonetheless, the methods are not detailed given the relatively short paper with two bibliographic entries only. Last but not least, Kim et al. (2020) trained a CNN on the DoS instance of the CIC-IDS2018 coupled with the KDDCup99. The main disadvantage of this approach is the use of an outdated dataset which dates back to the year 1999.

Table 4.6: Comparison of performance scores with similar work based on CIC-IDS2018

	Accuracy	Precision	Recall	AUC	Graph-based
Atefinia and Ahmadi (2021)	100.00	100.00	100.00	N/A	No
Catillo et al. (2020)	99.20	95.00	98.90	N/A	No
D'hooge et al. (2020)	96.00	99.00	79.00	N/A	No
Fitni and Ramli (2020)	98.80	98.80	97.10	0.94	No
Kanimozhi and Jacob (2019)	99.97	99.96	100	1	No
Kim et al. (2020)	99.99	81.75	82.25	N/A	No
Our Method	**99.60**	**100**	**99**	**1**	**Yes**

In terms of accuracy, our results may not be the best but it is well known that accuracy alone is not be a good evaluation metric when the data is imbalanced. As for precision and AUC, we achieve the top possible score. Finally, the obtained recall in our experiments is considered the second best score compared with the literature.

In brief, it was found that most of the relevant studies use the accuracy metric. On the contrary, the AUC score is neglected and was only considered in these papers (Fitni and

Ramli, 2020; Kanimozhi and Jacob, 2019). This finding relates to the disproportional class distribution of the dataset. Therefore, the accuracy metric should be coupled with other metrics in order to provide a correct assessment. It is also worth noting that AUC is a more robust metric in such tasks as observed in several papers (Lee, 2019; Buda et al., 2018)

Chapter 5

Conclusion and Recommendations

This chapter concludes the work with a summary of the research findings and provides recommendations for future research.

5.1 Conclusion

The problem of detecting malicious hosts in a dynamic network has recently converged to many artificial intelligence applications. One important consideration in building any anomaly detection system is maintaining TPR and FPR at a tolerable level. Although many network-based approaches have shown acceptable detection capabilities, leveraging meaningful modeling methods like graphs could further enhance these capabilities. In this thesis, we adopt the sliding window model to the dynamic network and represent it a sequence of overlapping snapshot graphs to capture the trends between windows. We then train a deep neural network on a set of extracted graph-based and time-based features to discriminate host behavior.

Although a graph is a powerful representation of objects and the relationships among them, the detection of anomalous behaviors in graphs is challenging. One reason is the requirement of processing huge volumes of data. Another challenging problem is that most graphs (e.g., social network) can have billions of vertices and edges that appear and disappear over time. Moreover, finding a way to transform graph structure without manual feature extraction so that it can be leveraged by machine learning models is not straightforward. Also, determining the useful patterns that could be extracted from these graphs is one of the most challenging questions. The main goal of this thesis is to build an anomaly classifier based on deep neural networks for the discovery of malicious behaviors in the network. It

is also important to understand under which conditions the algorithm performs better and compare its efficiency against a well-known dataset. Finally, assessing the performance of the anomaly classifier using evaluation metrics associated with standard classification problems.

In Chapter 2, the research has provided a comprehensive description of the state-of-the-art methods for anomaly detection in graph-based data. Most anomaly detection methods are based on static graphs that only represent a single snapshot of the given data. These techniques could be useful for forensic analysis for example. Furthermore, they could be generalized to various applications, but normally result in a high false positive rates. However, real-life networks are constantly evolving, which demands a shift in research to dynamic graphs. Unlike static graphs, dynamic graphs are subject to structural changes when nodes or edges are added and deleted. More specifically, there are different types of anomalies in dynamic graphs as nodes and edges can be added or deleted at every timestep. Therefore, the research reviewed many machine learning approaches that predict and discover new patterns using data modeled as graphs. The main challenge, however, is to find a method to represent this high-dimensional data about graph structure to incorporate it into a machine learning model. Conventional approaches mainly rely on user defined features such as degree and neighbourhood statistics. Recent advances in representation learning, such as network embeddings and deep neural networks, allow us to capture trends and patterns.

Following, a description of the anomaly detection approach that was applied on the CIC-IDS2018 dataset (Sharafaldin et al., 2018) is detailed in Chapter 3. The method examined network graphs over time to determine which hosts are most likely to be malicious by generating graphs at regular intervals and extracting useful graph-based and time-related flow-based features. Then, a time-series is assembled based on the extracted features for each host. Finally, a deep neural network is trained to detect malicious nodes. The seven graph-based features used in this work are: in-degree, out-degree, betweenness centrality, eigenvector centrality, PageRank, authority and hub centrality. After computing the time series for each unique host during each interval, LSTM is leveraged to create our models because it is a good candidate for processing time-correlated sequence such as network traffic.

Finally, Chapter 4 compared the performance scores of stateful and stateless LSTM networks for anomaly detection. Moreover, a comparison of the results with varied window sizes and step sizes was carried out to demonstrate the impact of these hyper-parameters on performance. It was found that the accuracy is in the range of 93% to 94%, for stateless LSTM configuration. Moreover, the high recall-low precision negatively impacts the quality of the system and indicates that the classifier returns a lot of false positives. Furthermore,

the results indicate that the stateful LSTM configuration outperforms the stateless configuration where the best achieved accuracy is from the botnet scenario, at 100%. In addition to accuracy, the high recall and high precision values indicate that correctly identifies almost all malicious hosts as malicious. However, most related work neglected the AUC metric, which relates to the disproportional class distribution of the dataset. Hence, the AUC metric is considered the most robust performance metric, and in our work, the top possible scores of AUC and precision are achieved.

5.2 Research Limitations and Future Work

There are many limitations of both approaches. First, we will describe the common limitations and then describe the limitations identified with the stateless model. Manual feature extraction is time-consuming and prone to errors as it requires analysis by a subject matter expert. Recent graph embedding techniques can overcome this issue. Moreover, the selected dataset although considered latest but it is not up-to-date. Attackers establish different attack tactics very frequently, and therefore, the research community need to consider upgrading datasets more often. Also, the models were trained and tested on the same scenario of the dataset. It would be more realistic if we use one scenario for training and another for testing to see how the model would generalize. In addition, the research could be further extended to examine streaming graphs instead of only considering time-evolving graph. Last but not least, the research lacks time and computational analysis, which is a very crucial factor for real-world implementations and adoptions.

On the other hand, for the limitations of stateless model in specific, we can justify the degraded performance due to a couple of reasons. First, the selected training epoch (i.e., 200) had a major impact on the learning outcome. Ideally, more training epochs would be used (at least 1000), but this was truncated to 200 to maintain reasonable run times. Moreover, to further improve the performance, it is important to use an oversampling technique, such as the Synthetic Minority Oversampling Technique (SMOTE) algorithm (Fernández et al., 2018), to address the class imbalance problem by creating additional examples to the minority class.

References

Abello, J., Eliassi-Rad, T., and Devanur, N. (2010). Detecting novel discrepancies in communication networks. In *2010 IEEE International Conference on Data Mining*, pages 8–17. IEEE.

Agrawal, S. and Agrawal, J. (2015). Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48.

Akoglu, L. and Faloutsos, C. (2010). Event detection in time series of mobile communication graphs. In *Army science conference*, volume 1.

Akoglu, L., Tong, H., and Koutra, D. (2015). Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688.

Almutairi, S., Mahfoudh, S., Almutairi, S., and Alowibdi, J. S. (2020). Hybrid botnet detection based on host and network analysis. *Journal of Computer Networks and Communications*, 2020.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39.

Antonellis, I., Garcia-Molina, H., and Chang, C.-C. (2008). Simrank++ query rewriting through link analysis of the clickgraph (poster). In *Proceedings of the 17th international conference on World Wide Web*, pages 1177–1178.

Atefinia, R. and Ahmadi, M. (2021). Network intrusion detection using multi-architectural modular deep neural network. *The Journal of Supercomputing*, 77(4):3571–3593.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Becchetti, L., Castillo, C., Donato, D., Leonardi, S., and Baeza-Yates, R. A. (2006). Link-based characterization and detection of web spam. In *AIRWeb*, pages 1–8.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591.

Beutel, A., Xu, W., Guruswami, V., Palow, C., and Faloutsos, C. (2013). Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130.

Bhagat, S., Cormode, G., and Muthukrishnan, S. (2011). Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Buda, M., Maki, A., and Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.

Cao, S., Lu, W., and Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900.

Catillo, M., Rak, M., and Villano, U. (2020). 2l-zed-ids: A two-level anomaly detector for multiple attack classes. In *AINA Workshops*, pages 687–696.

Chamberlain, B. P., Clough, J., and Deisenroth, M. P. (2017). Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*.

Chen, H., Perozzi, B., Hu, Y., and Skiena, S. (2018). Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen, H.-H. and Giles, C. L. (2013). Ascos: an asymmetric network structure context similarity measure. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 442–449. IEEE.

Chollet, F. et al. (2015). Keras.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711.

Data61, C. (2018). Stellargraph machine learning library. https://github.com/stellargraph/stellargraph.

Davis, M., Liu, W., Miller, P., and Redpath, G. (2011). Detecting anomalies in graphs with numeric labels. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1197–1202.

Ding, K., Li, J., Bhanushali, R., and Liu, H. (2019). Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM.

Ding, Q., Katenka, N., Barford, P., Kolaczyk, E., and Crovella, M. (2012). Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

D'hooge, L., Wauters, T., Volckaert, B., and De Turck, F. (2020). Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. *Journal of Information Security and Applications*, 54:102564.

Fernández, A., Garcia, S., Herrera, F., and Chawla, N. V. (2018). Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905.

Fitni, Q. R. S. and Ramli, K. (2020). Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 118–124. IEEE.

Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., and Han, J. (2010). On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 813–822.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

Google Transparency Report (2020). Https encryption on the web. https://transparencyreport.google.com/https/overview. Accessed: 2020-10-22.

Goyal, P. and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94.

Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Hassanzadeh, R., Nayak, R., and Stebila, D. (2012). Analyzing the effectiveness of graph metrics for anomaly detection in online social networks. In *International Conference on Web Information Systems Engineering*, pages 624–630. Springer.

Haveliwala, T. H. (2003). Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, 15(4):784–796.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

Heard, N. A., Weston, D. J., Platanioti, K., Hand, D. J., et al. (2010). Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling

in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeh, G. and Widom, J. (2002). Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543.

Ji, T., Yang, D., and Gao, J. (2013). Incremental local evolutionary outlier detection for dynamic social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–15. Springer.

Jiang, M., Cui, P., Beutel, A., Faloutsos, C., and Yang, S. (2016). Catching synchronized behaviors in large networks: A graph mining approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(4):1–27.

Kahneman, D. (2011). Thinking, fast and slow (kindle edition).

Kanimozhi, V. and Jacob, T. P. (2019). Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *International Journal of Engineering Applied Sciences and Technology*, 4(6):2455–2143.

Kim, J., Kim, J., Kim, H., Shim, M., and Choi, E. (2020). Cnn-based network intrusion detection against denial-of-service attacks. *Electronics*, 9(6):916.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kleinberg, J. M. et al. (1998). Authoritative sources in a hyperlinked environment. In *SODA*, volume 98, pages 668–677. Citeseer.

Kopp, E., Kaffenberger, L., and Jenkinson, N. (2017). *Cyber risk, market failures, and financial stability*. International Monetary Fund.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kruegel, C., Mutz, D., Valeur, F., and Vigna, G. (2003). On the detection of anomalous system call arguments. In *European Symposium on Research in Computer Security*, pages 326–343. Springer.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

Kuang, D., Ding, C., and Park, H. (2012). Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 106–117. SIAM.

Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51.

Langville, A. N. and Meyer, C. D. (2005). A survey of eigenvector methods for web information retrieval. *SIAM review*, 47(1):135–161.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, J.-S. (2019). Auc4. 5: Auc-based c4. 5 decision tree algorithm for imbalanced data classification. *IEEE Access*, 7:106034–106042.

Li, J., Dani, H., Hu, X., and Liu, H. (2017). Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, pages 2152–2158.

Li, X., Li, Z., Han, J., and Lee, J.-G. (2009). Temporal outlier detection in vehicle traffic data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1319–1322. IEEE.

Li, Y., Huang, X., Li, J., Du, M., and Zou, N. (2019). Specae: Spectral autoencoder for anomaly detection in attributed networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2233–2236.

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031.

Lin, P., Ye, K., and Xu, C.-Z. (2019). Dynamic network anomaly detection system by using deep learning techniques. In *International conference on cloud computing*, pages 161–176. Springer.

Mahdavi, S., Khoshraftar, S., and An, A. (2018). dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE.

Maruhashi, K., Guo, F., and Faloutsos, C. (2011). Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 203–210. IEEE.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Müller, E., Sánchez, P. I., Mülle, Y., and Böhm, K. (2013). Ranking outlier nodes in subspaces of attributed graphs. In *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, pages 216–222. IEEE.

Murphy, K. P. (2013). *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.].

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976.

Noble, C. C. and Cook, D. J. (2003). Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114.

Pandit, S., Chau, D. H., Wang, S., and Faloutsos, C. (2007). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210.

Pei, J., Jiang, D., and Zhang, A. (2005). On mining cross-graph quasi-cliques. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 228–238.

Peixoto, T. P. (2014). The graph-tool python library. *figshare*.

Peng, Z., Luo, M., Li, J., Liu, H., and Zheng, Q. (2018). Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, pages 3513–3519.

Perozzi, B., Akoglu, L., Iglesias Sánchez, P., and Müller, E. (2014a). Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1346–1355.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014b). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Perozzi, B., Kulkarni, V., and Skiena, S. (2016). Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*.

Ranshous, S., Shen, S., Koutra, D., Harenberg, S., Faloutsos, C., and Samatova, N. F. (2015). Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247.

Rong, Y., Huang, W., Xu, T., and Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98.

Singar, A. V. and Akhilesh, K. (2020). Role of cyber-security in higher education. In *Smart Technologies*, pages 249–264. Springer.

Sricharan, K. and Das, K. (2014). Localizing anomalous changes in time-evolving graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1347–1358.

Statista (2020). Digital 2020 global digital overview. https://datareportal.com/reports/digital-2020-global-digital-overview. Accessed: 2020-06-07.

Sun, J., Qu, H., Chakrabarti, D., and Faloutsos, C. (2005). Neighborhood formation and anomaly detection in bipartite graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077.

Tyler, J. R., Wilkinson, D. M., and Huberman, B. A. (2003). Email as spectroscopy: Automated discovery of community structure within organizations. In *Communities and technologies*, pages 81–96. Springer.

Venkatesh, B., Choudhury, S. H., Nagaraja, S., and Balakrishnan, N. (2015). Botspot: fast graph based identification of structured p2p bots. *Journal of Computer Virology and Hacking Techniques*, 11(4):247–261.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242.

Vlajic, N., Andrade, M., and Nguyen, U. (2012). The role of dns ttl values in potential ddos attacks: what do the major banks know about it? *Procedia Computer Science*, 10:466–473.

Wang, L., Rege, M., Dong, M., and Ding, Y. (2010). Low-rank kernel matrix factorization for large-scale evolutionary clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1036–1050.

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., and Weinberger, K. Q. (2019). Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*.

Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Yu, W., Cheng, W., Aggarwal, C. C., Zhang, K., Chen, H., and Wang, W. (2018a). Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2672–2681.

Yu, Y., Yao, H., Wang, H., Tang, X., and Li, Z. (2018b). Representation learning for large-scale dynamic networks. In *International Conference on Database Systems for Advanced Applications*, pages 526–541. Springer.

Zhao, P., Han, J., and Sun, Y. (2009). P-rank: a comprehensive structural similarity measure over information networks. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 553–562.

Zheng, L., Li, Z., Li, J., Li, Z., and Gao, J. (2019). Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *IJCAI*, pages 4419–4425.

ملخص

يعد اكتشاف الانحرافات في مجموعة بيانات معينة خطوة مهمة في العديد من التطبيقات في مجال الأمن السيبراني؛ بما في ذلك كشف الاختراق والاحتيال وتحليل الشبكة الاجتماعية. تكتشف العديد من هذه الأساليب القيم المتطرفة من خلال تحليل البيانات القائمة على الرسم البياني. يجعل تحليل الرسوم البيانية التقاط العلاقات والمجتمعات ممكنًا، فضلاً عن الحالات المتطرفة. تتمثل ميزة استخدام الرسوم البيانية في أنه يمكن بسهولة نمذجة العديد من مشاهد الحياة الواقعية من خلال رسم بياني يلتقط هيكلها وتبعياتها المتبادلة. على الرغم من أن اكتشاف الشذوذ في الرسوم البيانية يعود إلى التسعينيات، إلا أن التطورات الحديثة في البحث العلمي استغلت أساليب الذكاء الاصطناعي لاكتشاف الانحرافات على الرسوم البيانية. تهدف الأطروحة إلى البحث في العديد من خوارزميات الذكاء الاصطناعي لاكتشاف السلوك المتطرف في نشاط الشبكات وفهم الظروف التي تؤدي فيها كل خوارزمية أداءً أفضل من خلال مقارنة كفاءتها ودقتها بناءً على إحدى مجموعات البيانات المعروفة. تم تدريب جميع النماذج واختبارها على مجموعة بيانات CIC-IDS2018 التي أنشأها المعهد الكندي للأمن السيبراني. وجد أن الدقة المحققة هي 99.6% باستخدام خاصية الكشف عن الشذوذ. على الرغم من أن معظم الأعمال ذات الصلة أهملت مقياس AUC ، والذي يتعلق بالتوزيع غير المتناسب لفئة مجموعة البيانات، إلا أن مقياس AUC يعتبر أقوى مقياس أداء، وفي عملنا، تم تحقيق أعلى الدرجات الممكنة من AUC .

University of Bahrain
Deanship of scientific Research

جامعة البحرين
عمادة البحث العلمي

(استمارة إيداع البحث)

اسم الباحث: سارة سعد الرميحي

عنوان البحث: أداء خوارزميات تعلم الآلة في اكتشاف الحالات المتطرفة في الرسوم البيانية الديناميكية للشبكة

سنة النشر: ٢٠٢٢/٢٠٢١

الكلية: تقنية المعلومات البريد الإلكتروني: 20122438@stu.uob.edu.bh

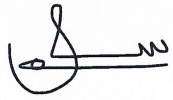
القسم: نظم المعلومات رقم الاتصال: ٣٦٩٣٦٣٢٨

إقرار الباحث بالموافقة على الشروط التالية:

يحق لجامعة البحرين إتاحة وتوفير الرسالة أو الأطروحة المذكورة أعلاه في مكتبة الجامعة أو من خلال موقع الحاوية العلمية للجامعة أو على نطاق أوسع في الخارج؛ كما ويمكن إعادة طباعة الرسالة إذا اقتضت الحاجة لذلك أو رقمنتها مع إتاحتها على شبكة الأنترنت وذلك حسب تقدير إدارة الجامعة.

تنطبق هذه الشروط على جميع النسخ بما في ذلك النسخ الإلكترونية

تعتبر هذه الرسالة أو الأطروحة من المواد المحمية والتي تخضع إلى حقوق النشر والتأليف، ولا يجوز نشر أي اقتباس من الرسالة بدون توثيق المصدر.

توقيع الباحث:  التاريخ: ٢٩ يناير ٢٠٢٢

جامعة البحرين

كلية تقنية المعلومات



أداء خوارزميات تعلم الآلة في اكتشاف الحالات المتطرفة في الرسوم البيانية الديناميكية للشبكة

أطروحة مقدمة من ضمن متطلبات الحصول على درجة الماجستير في الأمن الإلكتروني

مقدمة من :

اسم الطالب : سارة سعد الرميحي

الرقم الجامعي : ٢٠١٢٢٤٣٨

بإشراف :

د. هشام محمد العمال

(أستاذ مساعد)

جامعة البحرين

مملكة البحرين

يناير ٢٠٢٢