AN INTRUSION DETECTION SYSTEM USING RECURRENT NEURAL NETWORK WITH A REAL-WORLD DATASET

A MINI THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

OF

THE UNIVERSITY OF NAMIBIA

BY

HILMA NDAPEWA ALUDHILU

201403081

September 2020

SUPERVISOR: PROF R. R. PUENTE (COMPUTER SCIENCE DEPARTMENT, UNIVERSITY OF NAMIBIA)

ABSTRACT

Computer network technologies have grown speedily in the last decade and they are susceptible to numerous intrusions. Recently, Deep Learning methodologies for example Recurrent Neural Networks have been the new trend for building Intrusion Detection System (IDS). However, Recurrent Neural Network (RNN) IDS can be biased, as datasets from real-world production networks are mostly not used during the training of IDSs. The purpose of this study is to develop an IDS using RNN with two datasets from real-world networks and one synthetic dataset. The study aims to create a dataset from real-world network traffic, train an RNN model and test the corresponding IDS using several datasets: from real-world networks (Kyoto and UNAM datasets) and also synthetic (NLS-KDD dataset). Finally, the performance of the RNN-IDS is evaluated. A quantitative research design, with the experimental approach, is used for this study. An experiment is carried out to compare several RNN architectures: basic RNN, LSTM and GRU, offering insight into how the RNN architectures perform when trained and tested with real-world and synthetic datasets. The results of the study show that the GRU model outperformed other RNN architectures with an accuracy score of 95% using the real-world datasets and 97% using the synthetic dataset. The study concludes that the GRU model performs well with real-world datasets. Furthermore, the study recommends finding methods to solve the imbalance of classes in real-world datasets, without turning the data into synthetic data. It is also recommended to consider the environment in which the IDS will work in, before choosing the best model to be used.

LIST OF CONFERENCE PROCEEDINGS/JOURNAL ARTICLES

Aludhilu, H. N. (2018). *Critical Analysis of Intrusion Detection Approaches.* Paper presented at the Faculty of Science 6th Annual Science Research Conference 2018, University of Namibia, Namibia.

Aludhilu, H. N. & Puente, R. R. (2020). A Systematic Literature Review on Intrusion Detection Approaches. *Revista Cubana de Ciencias Informáticas*, *14*(1), 58-78.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FN - False Negatives

FP - False Positives

GRU - Gated Recurrent Unit

IDS - Intrusion Detection System

LSTM - Long Short-term Memory

NSL-KDD - Network Security Laboratory-Knowledge Discovery and Data Mining

RNN - Recurrent Neural Network

TN - True Negative

TP - True Positive

UNAM – University of Namibia

ACKNOWLEDGEMENTS

I would like to thank God for all the blessings and opportunities. My sincere appreciation goes to Nampower for the opportunity to pursue a Masters Degree at the University of Namibia. My deepest gratitude goes to my supervisor Prof R. R. Puente, for his guidance and encouragement. It was a great pleasure for me to conduct this work under his guidance.

My loving family and friends, thank you for the encouragement, support and constant love.

DEDICATION

This work is dedicated to my late grandmother, Kuku Julia Nuusiku yaFrans yUusiku.

Your life was a blessing, I thank God every time I remember you.

DECLARATIONS

I, Hilma Ndapewa Aludhilu, hereby declare that this study is my own work and is a true reflection of my research and that this work or any part thereof has not been submitted for a degree at any other institution.

Hilma Ndapewa Aludhilu		29 November 2019
Name of Student	Signature	Date

Chapter 1: Introduction

This chapter presents the background of the study, the research problem, statement of the problem, objectives of the study, significance of the study, limitations and the delimitations of the study.

1.1 Background of the study

Computer network technologies have grown speedily in the last decade and they are susceptible to numerous intrusions (Lotfallahtabrizi & Morgan, 2018). Network intrusion is the act of obtaining unauthorised access to a system, with the aim of compromising the system's security to steal or modify the information found on the system and compromise the availability, integrity and confidentiality of information on a system. Computer systems are exposed to security vulnerabilities and therefore the role of Intrusion Detection Systems (IDSs) has become important as they ensure network and information security (Yin et al., 2017), by detecting malicious activities on a network (Ponkarthika & Saraswathy, 2018b).

Recently, machine learning methodologies have been widely used for developing IDSs, with Deep Learning as a new trend for building IDSs (Ponkarthika & Saraswathy, 2018; Prajyot & Kalavadekar, 2018; Tang, Mhamdi, McLernon, Zaidi, & Ghogho, 2018; Yin et al., 2017). Machine learning methodologies have been effective in creating IDSs because they can automatically identify variations between normal data and abnormal data with a high accuracy. Machine learning methodologies allows

IDS to be developed with high detection rates and low false positive rates, the system adapt itself to possible changing malicious behaviours (Zamani & Movahedi, 2013). In addition, machine learning methodologies are able to detect unknown attacks as they have a strong generalizability (H. Liu & Lang, 2019).

According to Elsherif (2018), an IDS can be built using a Recurrent Neural Network (RNN) which is a Deep Learning approach. A Recurrent Neural Network-Intrusion Detection System (RNN-IDS) is able to identify seen and unseen threats, as it has a strong modelling ability, provides a highly accurate detection rate, and low false-positive rate, especially when using NSL-KDD dataset (Zamani & Movahedi, 2013). However, RNN-IDS tend to be biased, as datasets from real-world production networks are mostly not used (Elsherif, 2018) during the training process. Often, only benchmark datasets are used to train the IDS, although the benchmark datasets are not strong representatives of real-world traffic. (Bhuyan et al., 2015). This increases the need to use datasets from real-world networks when developing RNN-IDSs, for them not to be biased in the real-world environment in which they will operate.

1.2 Statement of the Problem

The lack of traffic datasets from a real-world production network is a challenge that faces the development of an effective and unbiased IDS. Intrusion detection systems including RNN-IDSs tend to be biased in real-world environments as less or no datasets from real-world networks are used to train and test models when developing the IDSs (Elsherif, 2018). Therefore, data from real-world networks need to be used

Goes, 2013). The limitation of this study is that all possible network intrusions might not be presented in the dataset created from the real-world networks.

1.6 Delimitation of the study

1.7 Delimitations of the research are characteristics that determines the limits of the study (Simon & Goes, 2013). This study confined itself to developing and evaluating the performance of an RNN-IDS model, based on the main functionality of an IDS: to detect attacks on a network. It also confines itself to working only with three Recurrent Neural Network architectures: basic RNN, LSTM and GRU.

1.8 Thesis Outline

This section outlines the structure of the thesis according to the chapters.

Chapter 1: Introduction, it is the first chapter of the thesis which serves as an introduction to the study. It presents the background of the study, research problem, statement of the problem, objectives, significance, limitations and the delimitations of the study.

Chapter 2: Literature Review, presents the literature from different studies. The topics covered include the overview of Intrusion Detection Systems, Recurrent Neural Network for Intrusion Detection, real-world datasets, performance evaluation metrics of IDS models and finally a summary of related work based on the use and performance RNN IDS.

Chapter 3: Research Methodology, covers information regarding the research approach, research design (Experimental Design), research instruments, data collection procedure, data analysis and research ethics considered in the study.

Chapter 4: Results and Discussions, presents the results and further discuss the results of the study, focusing on the creation of a traffic dataset from a real-world network, training and testing an RNN-IDS using several datasets and finally the performance evaluation of the best model.

Chapter 5: Conclusions, provides the conclusions which arise from the study, in connection with the objectives of the study.

Chapter 6: Recommendations, presents the recommendations from the study, providing suggestions regarding the imbalance of classes in real-world datasets and training of RNN models using real-world datasets.

Chapter 2: Literature Review

In this chapter, the literature from different studies related to the research topic is presented. The topics began by covering an overview of Intrusion Detection Systems, Recurrent Neural Network for Intrusion Detection, traffic datasets from real-world networks for intrusion detection, performance evaluation metrics of IDSs. Finally, the chapter presents a summary of related work based on the use and performance of RNN IDS.

2.1 Intrusion detection systems

Intrusion detection has become a substantial topic in network security, as network threats have increased significantly. Intrusion detection systems nowadays have become a basic security measure in computer networks (Yin, Zhu, Fei, & He, 2017) and therefore it is of great importance. Sarmah (2001, p.4) defined an intrusion detection system (IDS) as "a security system that regulates computer systems and network traffic and analyses that traffic for potential attacks from outside the organisation, as well as for system attacks from inside the organisation". An IDS is also defined by Xu, Shen, Du, and Zhang (2018) in simple terms as a security management system used to detect network intrusions. The purpose of using an IDS is to protect the integrity, confidentiality and availability of the data (Kadam & Deshmukh, 2014) and systems from intruders such as unauthorized users trying to carry out a task in the system (Mylavarapu et al., 2015).

The main functions of IDS include monitoring the behaviours of a user, discovering and responding to suspicious activities and also reporting suspicious activities to the

system administrator. An IDS aims to detect intrusions in real-time and respond to the intrusions accordingly before the intruder gets hold of confidential information or causes harm to the system. Desirable properties of an IDS includes minimal human control, high accuracy, with la ow false alarm rate should be low, able to detect the attacks and respond quickly (Choudhary & Swarup, 2009; Richariya, Singh, & Mishra, 2012).

2.1.1 Two major types of Intrusion Detection Systems

There are two major types of IDS: anomaly-based IDS and signature-based IDS. Anomaly-based IDS relies on the network behaviour (Jyothsna & Prasad, 2019) and it is designed to understand and differentiate between the normal (non-malicious) and the malicious behaviour of the system (Satam, 2015). The development of an anomaly-based IDS includes the training and testing phase. During the training phase, the model learns both malicious and non-malicious traffic, and then, during the testing phase, the data set is used to determine the IDS's capability to detect intrusions (Khraisat et al., 2019). The design of the anomaly-based IDS starts with the collection of data consisting of non-malicious behaviour for the network and malicious behaviour for the network (Satam, 2015). Anomaly-based IDSs have benefits such as capabilities of discovering internal malicious activities and making it difficult for a system intruder to identify a normal user (Khraisat et al., 2019). The limitation of the anomaly-based IDSs is that they can obtain a high false-positive rate and also a low detection rate (Yeo et al., 2017).

Signature-based intrusion detection system makes use of the knowledge obtained from analysing the behaviour of known intrusions on the network and the system looks for signs of previously known intrusions from the database of intrusion signatures (Satam, 2015). Signature-based IDS relies on pattern matching techniques to detect known attacks (Khraisat et al., 2019). Intrusions in the signature-based IDS are detected by matching their signatures with intrusion signatures in the signature database. The advantages and disadvantages of the anomaly-based and signature-based IDSs are highlighted in Table 1.

Table 1: Advantages and Disadvantages of Anomaly-based IDS and Signature-based IDS

	Advantages	**Disadvantages**
Anomaly-based IDS	- Detects zero-day attack attempts. - Identifies intrusions with low false rates. - Detect new intrusions. - Creates intrusion signatures.	- Obtains a low detection rate for the known attacks. - Gives a high false-positive rate. - Needs initial training.
Signature-based IDS	- Gives a high response time for known attacks. - Detect intrusions with a low false-positive rate.	- The signature database needs to be updated frequently. - Limited capability to detect zero-day attacks - Designed to detect attacks for known attack signatures only. Therefore, it cannot detect new attacks.

2.1.2 Five major types of attacks detected by Intrusion Detection Systems

IDSs are capable of detecting different types of network attacks. An attack is a security threat that involves, but it is not limited to, attempting to steal, obtain and alter

information without authorized access or gain access to a network without permission (Agrawal et al., 2017). An attack has characteristics that can compromise information or a network. The four major categories of attacks are Denial of Service (DoS), User to Root (U2R), Remote to User (R2L) and Probing (Ahmed, Mahmood, & Hu, 2016; Agrawal et al., 2017). The aforementioned attacks are described below:

- Denial of Service attack is when the attacker prevents the authorised user from using the computer services (Dias et al., 2017). DoS can be done by flooding the network and disrupting a connection or service. Examples of DoS attacks include Smurf, Land and Ping of Death (Pod).

- The user to root attack takes place when the attacker gets access to a normal user account and then exploits the system vulnerabilities to gain root privileges (Dias et al., 2017). Examples of U2R attacks are Loadmodule, Rootkit and Bufferoverflow.

- Remote to User attack is when the attacker gets unauthorised access to a local machine to send packets over the network. The Remote to User attack allows the attacker to have privileges which the authorised user normally have when using the computer. Examples of Remote to User attack are Ftp_write, Warezclient and Imap.

- A probe is a program that automatically scans and monitors the network activities and collects data from the network. The attacker collects information about the network and also finds vulnerabilities which can be used to attack the

network. Probing attacks include Ipsweep, Nmap and Portsweep (Dias et al., 2017).

2.2 Machine Learning Intrusion Detection Techniques

Machine learning is a section of artificial intelligence which involves the design and development of algorithms (Jha & Ragha, 2013). Additionally, Machine learning is the study of algorithms which are able to improve their performance using their experience (Hamid et al., 2016). One of the focus of machine learning is to automatically learn to identify complex patterns and make intelligent data-based decisions (Jha & Ragha, 2013).

Machine learning methodologies have been widely used to develop IDS to detect various types of attacks (Yin et al., 2017). Equally important, Machine learning techniques can also help the network administrator correspond accordingly to prevent intrusions. Numerous machine learning techniques such as Artificial Neural Network, Decision Trees, Support Vector Machine, Genetic Algorithm and Bayesian Networks are used for the implementation of IDS. The five different machine learning techniques as outlined as follows:

- Support vector machines (SVM) is a classification method (Jha & Ragha, 2013), which is able to perform different classification tasks, analyse data and recognize patterns (Chowdhury et al., 2016). Furthermore, SVMs are used to implement IDS which are able to provide real-time detection of intrusions (Shah, Hayat, & Awan, 2015).

- Fuzzy logic can be used in anomaly IDS as it deals with decision making and reasoning (Shah et al., 2015). Moreover, used for anomaly detection as they enable an object to belong to various classes simultaneously, making it possible for detecting intrusions (Singh & Nene, 2013a).
- A Decision Tree is a simple powerful classification algorithm (Shah et al., 2015) used for classification problems (Singh & Nene, 2013a). According to Rai, Devi, and Guleria (2016), a Decision Tree consists of internal nodes representing test on an attribute, branches which are outcomes of the test and leaf nodes which are class labels. A Decision Tree is powerful at classification, it is used to implement IDSs which are able to classify and detect intrusions.
- Genetic Algorithm is a search method or optimization technique that is based on the genetic principle and natural selection (Wang, Yang, & Ren, 2009). According to Sharma and Nema (2013), Genetic Algorithm is capable of producing high-quality IDS solution by applying the concept of selection and evolution. Genetic Algorithm has been recently used to support IDSs, by creating new rules from existing rules (Sharma & Nema, 2013).
- Bayesian networks are augmented directed acyclic graphs with vertices and directed edges (Devarakonda et al., 2012). Bayesian networks are able to provide competitive results when it comes to detecting intrusions because of its structure. Dhopte and Chaudhari (2014) also support that the Naive Bayesian Network has certain properties that make an IDS useful and accurate.
- Artificial Neural Network is a classification technique (Shah et al., 2015), which can be used to detect intrusions. According to Patel et al., 2015, the key

goal of Neural Network approach to intrusion detection is to acquire the behaviour of different users of the system. Artificial Neural Network IDSs can be implemented using deep learning. Deep learning-based intrusion detection system model is currently being proposed (Ponkarthika & Saraswathy, 2018). Deep learning is an advanced sub-field of machine learning which can be used to extract higher degrees of relationships among data (Elsherif, 2018).

Currently, the Recurrent Neural Network and Convolutional Neural Network are the main models of deep learning algorithms which can be used to build IDSs (Elsherif, 2018). According to Ponkarthika and Saraswathy (2018), Convolutional Neural Network increases the accuracy of intrusion detection when classifying threats, through improved behavioural features. Several studies (Liu, Liu, & Zhao, 2017; Vinayakumar, Soman, & Poornachandran, 2017; Mohammadpour, Ling, Liew, & Chong, 2018) have indicated that Convolutional Neural Network for IDSs needs future work such as improving false alarm rate and the quantity of normal data learning, in order to decrease false alarm rate. In addition, providing real data for learning and testing. Furthermore, a study by Vinayakumar, Soman, and Poornachandran (2017) identified that the Convolutional Neural Network requires more computational cost because of its complex architecture. Recurrent Neural Network (RNN) can be used for supervised classification learning and has the potential to generalise information used to distinguish threats which are seen and unseen in IDS (Mohammadpour et al., 2018). Additionally, the RNN also has a good modelling capability for intrusion detection with high accuracy and detection rate and a low false positive rate, especially when it comes to the classification of the NSL-KDD dataset (Lin et al., 2018). In detecting

intrusions, RNNs tend to outperform other models such as Convolutional Neural Network with high accuracy percentages and this is because the CNNs are mostly designed and used for image processing applications (Vani, 2017). The strengths and limitations of the five different machine learning techniques are outlined in Table 2.

Table 2: Strengths and Limitations of Machine Learning Techniques

Machine Learning Technique	**Strengths**	**Limitations**
Neural Network	- Does not need expert knowledge and it is able to detect unknown or new intrusions (Shah et al., 2015). - Flexible, fast and capable of analysing non-linear data set with multiple variable (Shah et al., 2015). - Neural network is able to make decisions quickly and detects intrusions in real-time.	- Over-fitting may happen during neural network training (AliShah, Sikander Hayat Khiyal, & Daud Awan, 2015).
Bayesian Network	- Make use of both prior knowledge and data (AliShah et al., 2015).	- May not carry out correct classification if the prior knowledge happens to be wrong (AliShah et al., 2015).

Support Vector Machine	- Has a better learning ability, especially for small samples (Singh & Nene, 2013; AliShah et al., 2015) - Learn a larger set of patterns which allows it to be able to scale better (Jha & Ragha, 2013). - Able to update the training patterns dynamically when there is a new pattern during classification (Jha & Ragha, 2013).	- Mostly uses binary classifier which is not able to give further information about the type of attack which is detected (Singh & Nene, 2013) - It requires the processing of raw features for classification, increasing the complexity of the architecture and reduces the intrusion detection accuracy (Jha & Ragha, 2013).
Genetic Algorithm	- Capable of solving numerous practical classification problems, especially the problems which are non-linear and they involve small samples (Singh & Nene, 2013).	- Genetic algorithm does not guarantee constant optimization of the response times (AliShah et al., 2015).
Fuzzy Logic	- Effective at detecting port scans and probes (AliShah et al., 2015)	- The operation of the Fuzzy logic involves a high resource consumption (AliShah et al., 2015)

Decision Tree	- Decision Tree works well with a large number of datasets (Singh & Nene, 2013; AliShah et al., 2015). - It provides a high detection accuracy (Jha & Ragha, 2013; AliShah et al., 2015). - Adapted in a fast manner (Jha & Ragha, 2013). - Works well in real-time intrusion detection because Decision Tree offers a high detection performance and can create and interpret a model easily (Singh & Nene, 2013).	- Classification accuracy is drastically reduced when there are too many categories to be classified (Wang et al., 2009).

As outlined in Table 2, different Machine Learning techniques used for intrusion detection have their strengths and limitations. Based on the strength of the Machine Learning techniques, Neural Networks was identified to have favourable characteristics for implementing IDSs. Apart from being able to learn, Neural Networks have the ability to make decisions quickly and detect intrusions in real-time, providing high accuracy. This is a strong characteristic in detecting intrusions as intrusions are expected to be detected in real-time to prevent attackers from causing harm to the system. The neural network also does not need expert knowledge, meaning it needs minimum human intervention in order to detect intrusions. The Recurrent Neural network specifically offers a high accuracy and detection rate, together with low false-positive rate and therefore can be considered ideal for building IDS with the aforementioned qualities that the RNN offers.

2.3 Recurrent Neural Network for Intrusion Detection

Recurrent Neural Network (RNN) is an expansion or type of neural networks which has an internal loop (Satam, 2015) as illustrated in Figure 1. The internal loop of the RNN is used to process sequential information (Scikit-learn, 2019). Additionally, the loop links each layer, stores and memorises the previous input and apply it to the current output (Ponkarthika & Saraswathy, 2018).

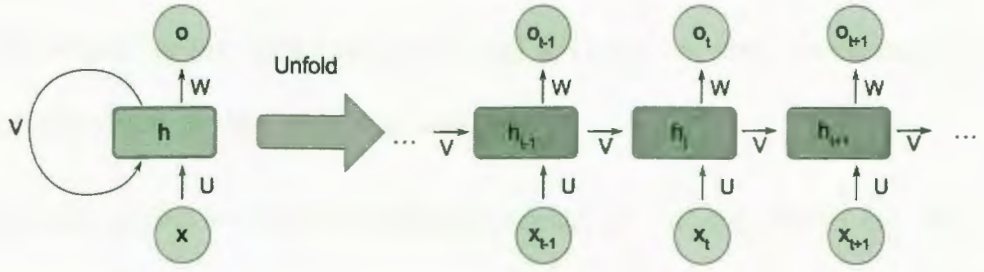

Figure 1. Simple RNN architecture (Tiwari, 2018)

One of the key features of RNN is the ability to circulates information in a hidden layer which can recall previously processed information (Xu et al., 2018). In the RNN, the hidden unit component remembers or forgets certain information, which is important for processing time-series information (Xu et al., 2018). In the same manner, several intrusion behaviours can be derived from the network as time series of events from the network. Therefore, RNNs are considered to be suitable for the development of IDSs (Xu et al., 2018).

Recurrent Neural Network (RNN) can also be used for supervised classification learning and is capable of generalising seen and unseen threats in IDS (Tiwari, 2018). Therefore, Recurrent Neural Networks are used to build IDSs as they perform well in classification problems (Elsherif, 2018). Additionally, it is observed that the RNN also has a strong modelling ability for intrusion detection with high accuracy and detection rate and a low false-positive rate, especially when it comes to the classification of the NSL-KDD dataset (Lin et al., 2018). In detecting intrusions, RNNs specifically offers a high accuracy and detection rate, together with low false-positive rate and therefore

can be considered ideal for building IDS with the aforementioned qualities that the RNN offers. On the other hand, traditional RNN suffers from the vanishing gradient problem which can lead to less accuracy.

Recurrent Neural Network has different variations which include Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTM is a type of recurrent neural network capable of learning to carry out sequence prediction (Elsherif, 2018). The LSTM unit consists of the input gate, forget gate, output gate, and the memory cell as shown in Figure 2. The cell remembers values over time intervals, the input gate determines the input ratio, the forget gate passes the previous memory and the output gate determines if the output of memory cell should be passed or not (Ponkarthika & Saraswathy, 2018b).

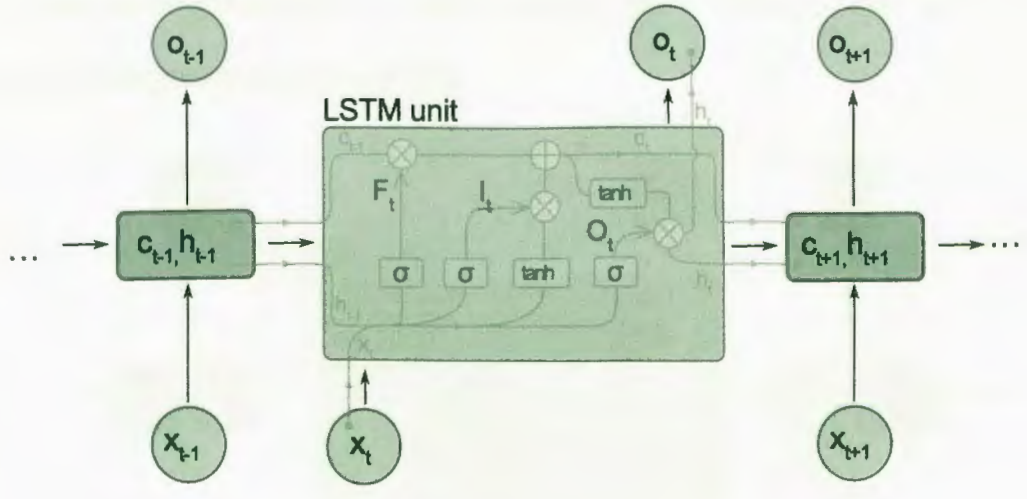

Figure 2. LSTM architecture (Deloche, 2017b)

The main goal of LSTM is to obtain a disappearing gradient descent to prevent problems of long-term dependency (Althubiti et al., 2018). Furthermore, the LSTM is

regarded by Le, Kim and Kim (2019) capable of long-term learning as it solves the vanishing gradient problem, compared to the traditional RNN model, and therefore, the LSTM is used to detect anomalies (Althubiti et al., 2018) like the ones which could exist on a network.

Another variation of RNN is the GRU architecture, which is considered as a simplification and enhancement of the LSTM (Xu et al., 2018). The GRU architecture shown in Figure 3 integrates the forget and input gate, to have a single gate referred to as the update gate and it also combines the cell state and hidden state (Le et al., 2019). A GRU is known to have two gates; a reset gate which describes the combination of the new input and previous memory and an update gate which determines the amount of previous memory to be kept. This means that to make predictions, the GRU employs the gated mechanisms which is used to regulate input, memory, and the currently available information (Li et al., 2018).

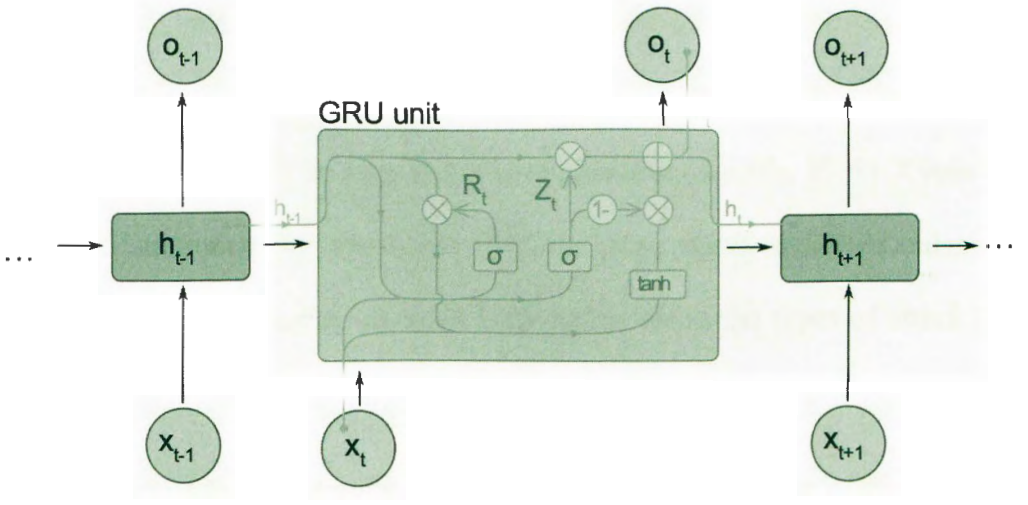

Figure 3. GRU architecture (Deloche, 2017a)

2.4 Network traffic dataset from a real-world network environment

The network traffic datasets are an important aspect of anomaly-based intrusion detection systems. Traffic dataset which can be used for training IDSs can be from real-world environments. In this context, the environments are considered to be real because the network traffic is captured from a productive network environment (Ring et al., 2019). Traffic datasets from real-world networks present the existing real networks moderately, compared to synthetic datasets such as NLS-KDD (UNB, 2015), which according to Protic (2018) is not ideal for the portrayal of current real networks. Additionally, the real-world traffic content can be considered not to be biased compared to synthetic traffic content which may be biased (Cermak et al., 2018).

An example of a dataset from a real-world environment is the Kyoto University dataset (Takakura, 2015). The Kyoto 2016 dataset is a collection of the university network traffic data built on real traffic data and captured and obtained using honeypots, darknet sensor, e-mail server and web crawler (Protic, 2018). The Kyoto University honeypot systems' network traffic data consists of 24 statistical features, 14 features from the KDD Cup 1999 dataset, and 10 more features (Agarap, 2018). Kyoto dataset consists of data from existing real networks, consisting of non-malicious and malicious data, but it does not mention detailed information about the types of attack (Protic, 2018).

There are several limitations of using datasets from real-world network traffic. These datasets are mostly imbalanced as they contain more non-malicious user behaviour compared to malicious user behaviour (Ring et al., 2019). Datasets which exhibit an

unequal distribution between its classes are described as an imbalanced dataset (Armah et al., 2014). When the data is imbalanced, the target class which is less is called the minority class, and the class which is more is called the majority class. Additionally, real-world network traces often need to be anonymised and the dataset from the real-world network does not mention information about the attack types which are present in the dataset (Protic, 2018). There are also concerns about real-world network traffic, such as integrity and cost (Haider et al., 2017).

2.5 Performance evaluation metrics of the model

The efficiency of a model can be measured using different metrics. The metrics include the accuracy, precision and recall, which can be calculated using a confusion matrix (Ingre & Yadav, 2015). A confusion matrix is known as a techniques used for measuring the performance of Machine Learning classification and it is also a comparison summary of the predicted and actual results in classification problems. The confusion matrix is in the form of a table with four different variations of the expected values and actual values as shown in Table 3.

Table 3. Confusion Matrix

		Actual Values: Negative	Actual Values: Positive
Predicted Values	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Additionally, the confusion matrix is comprised of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP refers to the number of malicious records which are correctly classified, TN refers to the number of non-malicious records which are correctly classified, FP refers to the number of non-malicious records which are incorrectly classified and FN refers to the number of malicious records which are incorrectly classified as not malicious (Ponkarthika & Saraswathy, 2018). The comparison summary provided by the confusion matrix is greatly important as it helps to determine the performance of the model after it is trained. The confusion matrix values are used to calculate the Accuracy, Precision,

Recall and F-measure metrics, that are used to evaluate the performance of models. The evaluation metrics are further explained in sub-sections 2.4.1-2.4.4

2.5.1 Accuracy

The Accuracy of the model is calculated by the ratio between the dataset samples which are correctly classified to the total amount of samples present in the dataset (Elsherif, 2018). The accuracy metric provides the percentage of true detection over the total traffic records, calculated using equation (1).

$$= \frac{\quad + \quad}{+ \quad + \quad +} \tag{1}$$

Accuracy is regarded to be critical in classification and the aim should be to achieve the highest accuracy (Yadav & Shukla, 2016). In addition, Prajyot and Kalavadekar (2018) also assert that accuracy is the significant performance indicator of an RNN model.

2.5.2 Precision

Precision is a metric which shows how many records, predicted by a model as intrusions, are actual intrusions. The precision is the ratio of true TP over the sum of TP and FP records (Elsherif, 2018) and it is calculated using equation (2).

$$= \frac{\quad\quad}{+} \tag{2}$$

2.5.3 Recall

Recall is the number of attack detection which are correct and it is also referred to as the True Positive Rate. It is the ratio of true positive over the sum of true positive and false negative records (Elsherif, 2018). The recall is calculated using equation (3).

$$= \frac{\quad}{+} \tag{3}$$

2.5.4 F-Measure

F-Measure is the mean of the Precision and Recall (Elsherif, 2018) which combines the properties of the recall and precision metrics. F-measure can be calculated using equation (4).

$$- \quad = \frac{2}{\frac{1}{\quad} + \frac{1}{\quad}} \tag{4}$$

When both the Precision and Recall reaches 100%, the F-Measure is knowm to be at maximum, which means that the classifier did not get any false alarms and detected all of the attacks.

Accuracy is the most common model evaluation metric, but it can be deceptive if the data are imbalanced. In such instances, it is needed to consider using other evaluation metrics, in addition to the accuracy metric. Even with the use of other matrices, a study by (Akosa, 2017) suggests that the imbalanced data can cause the evaluation metrics to disclose more about the class the distribution instead of the actual output of models

when the data are imbalanced. Therefore, when selecting a model, a combination of measures should be considered instead of only depending on one measure.

2.6 Related work on RNN IDS

Prior studies have modelled and evaluated the performance of IDSs based on the RNN Deep Learning approach. Yin et al. (2017) explored how to model an intrusion IDS based on deep learning using RNN. In their study, the RNN-IDS was proposed and was compared to other machine learning methods. The important performance indicator for their model was the accuracy of intrusion detection, which was used to measure the performance of the RNN-IDS model. The experimental results showed that RNN-IDS obtains high classification accuracy as the experiment showed a good detection rate of 83.28% on 100 epochs for the KDDTrain+ dataset (UNB, 2015). The study only used a basic RNN model and therefore recommends future classification performance using other RNN model types such as LSTM RNN and Bidirectional RNNs in order to review the classification efficiency of LSTM, Bidirectional RNNs algorithm for intrusion detection.

Elsherif (2018) also built an IDS system based on RNN Deep Learning architecture with bi-directional RNN, LSTM and bi-directional (LSTM) variations. The three RNN variations were tested using NSL-KDD (UNB, 2015) benchmark dataset which is a synthetic dataset. The performance of the proposed system was measured using the training accuracy, Recall, Precision and F-Measure. The results showed that Bi-Directional LSTM showed better results over other proposed RNN models.

Xu et al (2018) carried out an experiment using the KDD 99 and NSL-KDD data sets. The experiment obtained an detection rate of 99.42% using KDD 99 and a detection rate of 99.31% using NSL-KDD with low false-positive rates of 0.05% and 0.84%, respectively. This comparative experiment also showed that the GRU is more suitable for IDS, compared to LSTM, and has shown that the GRU is an effective enhancement of LSTM. The study proposes an IDS consisting of a RNN with GRUs.

Ponkarthika and Saraswathy (2018) modelled an IDS based on Deep Learning, using the basic RNN and LSTM, trained with Knowledge Discovery Dataset (KDD) Cup 1999 dataset (KDD Cup, 1999). The study verified that the Deep Neural Network is effective for IDS and the RNN was found to be an effective model to train sequence data. Additionally, the study also found that the attacks were detected well by the LSTM-RNN classifier which was used. Prajyot & Kalavadekar (2018) also described a Deep Learning technique for Intrusion Detection using RNN and trained the model using the NSL-KDD Dataset. The IDS model was found to improve the accuracy of intrusion detection.

The RNN-IDSs developed in the aforementioned studies mostly used the KDD Cup 1999 or NSL-KDD benchmark dataset and did not use datasets from real-world production networks to eliminate the possibilities of a biased IDS. This shows that there is a need to model and evaluate an unbiased RNN-IDS using a dataset from a real-world production network. Several variations of RNN also needs to be trained and tested using real-world data in order to compare their performance on the two different types of datasets. This study made use of datasets from real-world production networks

to train and test three variations of RNN and compared their performances to obtain the best model.

2.7 Summary

In this chapter, the literature from different studies related to this study was presented. The chapter provided an overview of IDSs, including different types of IDS such as the anomaly and signature-based, the types of major attacks detected by IDSs and the machine learning intrusion detection techniques. The chapter further presented the different RNN (Basic RNN, LSTM and GRU) which can be used for IDS, traffic datasets from real-world networks for intrusion detection, performance evaluation metrics of IDSs such as accuracy, precision, recall and f-measure. Finally, a summary of related work based on the use and performance RNN IDS was presented.

Chapter 3: Research Methods

This chapter covers information about the research methods. Section 3.2 presents the research approach, section 3.3 presents the research design, section 3.4 presents the research instruments, section 3.5 presents the research procedure, section 3.6 presents the data analysis and research ethics considered in this study are covered in section 3.7.

3.1 Research Approach

A research approach involves the steps regarding the methods of collecting, analysing and interpretating data (Chetty, 2016). This study used a quantitative research approach which emphasises on numbers and figures, therefore, regarded as being scientific (Eyisi, 2016). Studies about artificial neural networks are highly based on numbers and figures; therefore, a quantitative research approach is used in this study.

3.2 Research Design

A research design is a structure, plan or strategy of investigation, outlining steps to be followed when collecting and interpreting data in a study, to meet research objectives or answers to research questions (Kumar, 2014). For this study, an experimental research design which is mostly used in artificial neural network field is used. An experiment is carried out to train and test the RNN models using datasets which consist of network data from real-world network environments as well as synthetic data.

3.3 Research Instruments

A research instrument is a mean of gathering information for a study (Kumar, 2014). There are several data collection methods that can be used in quantitative research. This study used observations which were carried out during the training and testing of the RNN-IDS model.

3.4 Procedure

The procedure used to carry out this study involves the following steps:

1. Data collection
2. Data pre-processing
3. Experiment for training and testing RNN models
4. Performance evaluation of RNN best model
5. Analysis of results

Step 1 of data collection is presented in Section 3.4.1, followed by the data pre-processing which is presented in Section 3.4.2, Section 3.4.3 presents the experiment for training and testing RNN models, Section 3.4.4 presents the performance evaluation of RNN best model and finally, the analysis of results done in step 5 is presented in Section 3.4.5.

3.4.1 Data Collection

For the RNN models to be trained and tested, a dataset of network traffic is needed. The RNN models used in this study are trained and tested on three different datasets:

a dataset from the University of Namibia (UNAM) created as part of this research, a dataset from Kyoto University and a dataset from the Network Security Laboratory-Knowledge Discovery and Data Mining (NSL-KDD). The Kyoto and NSL-KDD datasets are available online for use and therefore they were downloaded. On the hand, permission needed to be sought from UNAM to collect network traffic to create a dataset to be used in this study.

Network traffics of the students and staff members of UNAM were captured using Wireshark (version 3.0.0). Wireshark is an open source network analyser software which captures live data flowing in the network interface and analyses the packets (Saxena, 2017). Wireshark uses the pcap library to capture packets and saves the data in a Packet Capture (pcap) file which was transformed to (Comma-separated values) csv file.

3.4.2 Data Pre-processing

After collecting the network traffic data from the UNAM network, the data were then pre-processed. Data pre-processing involved searching for missing values in the datasets. It is imperative to search if there were any missing values in the data because missing values in a dataset can cause the dataset to lose expressiveness, which leads to biased analysis and high possibility of wrong classification or predictions (Willems, 2017). The few rows found with missing values were excluded from the dataset.

The pre-processing of data also included converting qualitative or non-numeric data to numeric data. Some machine learning models such as RNN do not accept qualitative features and therefore the qualitative features needed to be converted to quantitative.

Qualitative to quantitative conversion is done by specifying a quantitative value for each qualitative value.

The following feature conversion and encodings were done: Conversion of source and destination addresses, conversion of time, the encoding of protocols and labelling of packets. The aforementioned feature conversion and encodings are outlined as follows:

- Source and destination addresses are converted to numeric values only, with no characters or letters. E.g. the address fe80::222d:7ff:fef7:63c5 was converted to a numeric value of 8022277635. During this conversion, each address was converted to a unique numeric value.
- Time is converted to a numeric value, for example 12:25:29 AM was converted to a numeric value of 0.01769.
- Encoding of protocols and encoding of information feature were done using numerical values as shown in Table 4 and Table 5 respectively.
- Packets are labelled with 0 indicating a packet to be non-malicious and 1 to be malicious. Malicious packets were found by using Wireshark suspicious traffic in Protocol Hierarchy statistics.

Table 4. Encoding of protocols

Protocol	Code
MDNS	1
ICMPV6	2
NTP	3
ARP	4
XID	5
UDP	6
TCP	7

Table 5. Encoding of information feature

Info	Code
Standard Query	1
ACK	2
Application Data	10

3.4.3 Experiment for training and testing RNN models

The experiment was implemented in a Python environment which was set up using TensorFlow and Keras Python libraries. Tensorflow is a Python open source software library used to develop and train Machine Learning models (TensorFlow, 2019) whereas Keras is a Python Deep Learning library. Keras is a Python high-level neural network API, which enables fast experimentation (Keras, 2019). In addition, Keras helps with easy implementation of deep neural networks, therefore it was used in the experiment environment of this study.

Pandas (Pandas, 2019), Numerical Python (NumPy) libraries, Matplotlib and Scikit-learn were also used in this study. Pandas is an open-source Python library which was used to import data and provide high-performance data manipulation. Pandas is also used as a data analysis tool. Pandas is built on NumPy which makes data manipulation and visualisation more convenient. NumPy is an open-source module of Python, which is used to create and provide mathematical computation on arrays (McKinney, 2012). Matplotlib (Hunter, 2007) is a plotting library for the Python programming language which was used in this study to visually plot the confusion matrix. Scikit-learn (Scikit-learn, 2019) is an open-source Python library built on NumPy, SciPy, and Matplotlib. Table 6 shows the software versions used in this study.

The study made use of a computer with the following specifications:

- Operating System: Mac OS High Sierra
- CPU: 2.5 GHz Intel Core i5
- RAM: 16GB DDR3
- GPU: Intel HD Graphics 4000 1536 MB

Table 6. Versions of software used

Software	Version
Python	3.6.8
Tensorflow	1.13.1
Keras	2.2.4
Pandas	0.24.1
NumPy	1.16.2
Scikit-learn	0.20.0
Matplotlib	3.0.2

3.4.3.1 Dataset Description

UNAM dataset, Kyoto 2016 dataset (Takakura, 2015), and NLS-KDD dataset (UNB, 2015) were used for the experiment in this study to train and test models. UNAM and Kyoto 2016 datasets consist of data extracted from real-world universities networks environments while NLS-KDD dataset consists of synthetic data. The NLS-KDD synthetic dataset was used in the experiment in order to compare its performance to the real-world datasets, using different models.

The UNAM dataset was created during this study and it consists of UNAM's network traffic data. The dataset was captured using Wireshark and the dataset consists of 8 features and a total of 55000 records. Also, the UNAM dataset consists of 54874 records regarded as non-malicious data and 126 records regarded as malicious. The

basic statistical characteristics of each numerical feature of UNAM dataset are shown in Figure 4. The descriptive statistics were used to assess the data by calculating the count, mean, standard deviation, minimum values, maximum values and quantiles of the data.

	0	1	2	3	4	5	6	7
count	55000.00000	55000.000000	5.500000e+04	5.500000e+04	55000.000000	55000.000000	55000.000000	55000.000000
mean	27500.50000	0.024832	2.985975e+14	1.921737e+10	3.522673	116.293418	4.644782	0.002291
std	15877.27674	0.007753	2.173104e+15	4.935703e+11	2.884208	128.371700	2.579217	0.047809
min	1.00000	0.000000	0.000000e+00	0.000000e+00	1.000000	42.000000	1.000000	0.000000
25%	13750.75000	0.022386	8.638000e+03	0.000000e+00	1.000000	60.000000	3.000000	0.000000
50%	27500.50000	0.025720	8.097065e+08	2.160000e+02	2.000000	70.000000	4.000000	0.000000
75%	41250.25000	0.029060	1.025530e+09	2.240025e+07	5.000000	111.000000	7.000000	0.000000
max	55000.00000	0.041670	8.087168e+16	8.049957e+13	23.000000	1514.000000	9.000000	1.000000

Figure 4. Basic statistical characteristics of UNAM dataset

A total of 8 features and a total of 55000 records from Kyoto dataset were used in this study. The Kyoto dataset consists of 48015 records regarded as non-malicious data and 6985 records were regarded to be malicious data. The basic statistical characteristics of each numerical feature of Kyoto dataset are shown in Figure 5.

	0	1	2	3	4	5	6	7
count	55000.00000	55000.000000	5.500000e+04	5.500000e+04	55000.000000	55000.0	55000.0	55000.000000
mean	27500.50000	0.219850	8.176779e+14	3.813821e+14	6.639436	0.0	0.0	0.127000
std	15877.27674	0.046799	5.443412e+15	1.124887e+15	0.535893	0.0	0.0	0.332976
min	1.00000	0.001440	1.153520e+09	1.153526e+09	6.000000	0.0	0.0	0.000000
25%	13750.75000	0.221920	1.150000e+13	1.150000e+13	6.000000	0.0	0.0	0.000000
50%	27500.50000	0.231230	1.150000e+13	1.150000e+14	7.000000	0.0	0.0	0.000000
75%	41250.25000	0.240870	1.150000e+15	1.150000e+14	7.000000	0.0	0.0	0.000000
max	55000.00000	0.970220	1.150000e+17	1.150000e+17	9.000000	0.0	0.0	1.000000

Figure 5. Basic statistical characteristics of Kyoto dataset

A combination of the Kyoto dataset and the UNAM dataset was also used to train the model. The Kyoto and UNAM datasets are combined to train and test the model using more data from two real-world network environment and also to generalise during the training of the model. The dataset consisted of 8 features and a total of 110000 records. A total of 107254 records were regarded as non-malicious data and 2746 records regarded to be malicious. The basic statistical characteristics of each numerical feature of Kyoto-UNAM dataset are shown in Figure 6.

	0	1	2	3	4	5	6	7
count	110000.000000	110000.000000	1.100000e+05	1.100000e+05	110000.000000	110000.000000	110000.000000	110000.000000
mean	27500.250000	0.122341	5.581377e+14	1.907007e+14	5.081045	58.146709	2.322345	0.064645
std	15876.988066	0.103117	4.152561e+15	8.179484e+14	2.594507	107.799083	2.952865	0.245900
min	1.000000	0.000000	0.000000e+00	0.000000e+00	1.000000	0.000000	0.000000	0.000000
25%	13750.750000	0.025670	8.097065e+08	2.160000e+02	2.000000	0.000000	0.000000	0.000000
50%	27500.000000	0.036000	1.921682e+11	3.424821e+10	6.000000	21.000000	0.500000	0.000000
75%	41250.000000	0.231230	1.150000e+13	1.150000e+14	7.000000	70.000000	4.000000	0.000000
max	55000.000000	0.970220	1.150000e+17	1.150000e+17	23.000000	1514.000000	9.000000	1.000000

Figure 6. Basic statistical characteristics of Kyoto-UNAM dataset

The NLS-KDD dataset consisting of 23 features and a total of 55000 records were used. The dataset consists of 29225 records which are regarded as non-malicious data and 25775 records are regarded to be malicious data. The basic statistical characteristics of each numerical feature of NLS-KDD dataset are shown in Figure 7.

	0	1	2	3	4	5	6	7
count	55000.000000	5.500000e+04	5.500000e+04	55000.000000	55000.000000	55000.000000	55000.000000	55000.000000
mean	296.300127	4.856340e+04	3.143764e+03	84.660818	27.703873	0.287306	0.285127	0.119325
std	2647.965511	5.873589e+06	7.793869e+04	114.731612	72.406857	0.447905	0.448337	0.319606
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000e+00	0.000000e+00	2.000000	2.000000	0.000000	0.000000	0.000000
50%	0.000000	4.400000e+01	0.000000e+00	14.000000	8.000000	0.000000	0.000000	0.000000
75%	0.000000	2.760000e+02	5.190000e+02	144.000000	18.000000	1.000000	1.000000	0.000000
max	42862.000000	1.167519e+09	5.153771e+06	511.000000	511.000000	1.000000	1.000000	1.000000

Figure 7. Basic statistical characteristics of NLS-KDD dataset

3.4.3.2 Training and testing different RNN models

The different RNN models (basic RNN, LSTM and GRU) were trained and tested with UNAM dataset, Kyoto dataset, a combination of UNAM and Kyoto real-world datasets and NLS-KDD synthetic dataset. The dataset for this study is small for a machine learning task, therefore, the k-fold cross-validation resampling procedure was used to fully utilise the data. The use of k-fold cross validation eliminates biases in choosing the appropriate RNN model to be used. Cross-validation is a technique that makes use of all available records as training and test examples (Bengio & Grandvalet, 2004). K-fold cross-validation involves splitting the data into K equal partitions, and then training a model on K – 1 partition while evaluating the model on the remaining partition (Chollet, 2018). The k-fold cross-validation technique (with k=10) was used to divide the dataset into a training set and testing set. The k-fold cross-validation technique was also used to train the models K times with a fraction 1/K of training data reserved for testing. In this study, data was split into several 10 subsets, called folds and then the model was trained and evaluated 10 times. The iterative process of

changing folds allows the dataset to be used as a training and a testing set during the iteration process. The data was split into folds as depicted in Figure 8.

```
Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [   17    28    30 ... 54938 54967 54991]

Training Set:  [    0     1     3 ... 54995 54996 54999]
Testing Set:   [    2     8    12 ... 54983 54997 54998]

Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [   37    39    43 ... 54942 54955 54961]

Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [    3    19    45 ... 54934 54957 54970]

Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [   21    24    32 ... 54988 54993 54995]

Training Set:  [    0     2     3 ... 54997 54998 54999]
Testing Set:   [    1     9    15 ... 54963 54977 54980]

Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [    7    16    23 ... 54985 54986 54990]

Training Set:  [    0     1     2 ... 54997 54998 54999]
Testing Set:   [    5    11    18 ... 54992 54994 54996]

Training Set:  [    1     2     3 ... 54997 54998 54999]
Testing Set:   [    0     4    20 ... 54972 54987 54989]

Training Set:  [    0     1     2 ... 54996 54997 54998]
Testing Set:   [    6    10    14 ... 54973 54979 54999]
```

Figure 8. Data split into 10-folds

During the training and testing, the batch size of 100 was used and the number of iterations called epochs was set to 100. The different RNN models are designed with the shape of the weight and bias set to random, with layers having a total unit of 50. Forward propagation was used for the RNN models.

To train the RNN models, a cost function "sigmoid cross-entropy with logits" was used as a binary classification was done. The Adaptive Moment Estimation (Adam) optimiser was used since it has fairly better performance, efficient computation, fewer memory requirements, easy implementation (Kingma & Ba, 2014). A loss function is specified to evaluate the weights and a logarithmic loss was also used. The loss function is defined in Keras as "binary cross-entropy" since the problem is a binary classification. At the end of the training, the accuracy and the recall score of the models were calculated. This means that the models' performance was evaluated according to their accuracy and recall score. The accuracy and recall scores evaluate the performance of the models because the accuracy is regarded as an important measure for evaluating the performance of the model (Yadav & Shukla, 2016) and the recall is important as the model for intrusion detection needs to be sensitive to detect positive instances. Although accuracy plays a major role in performance evaluation of models, using the accuracy score alone as the main performance metric is not enough, especially when there are severe class imbalances in the dataset (Tompkins, 2019). Therefore, the recall score was also used to evaluate the performance of the models, as the real-world datasets used in this study consisted of severe class imbalances.

3.4.4 Performance evaluation of the best model

After training and testing the models, the model with the highest accuracy score and recall score was evaluated. The performance of the best model was further evaluated using the precision, and F-measure score matrices.

3.5 Data Analysis

Data analysis is a process of systematically organising, examining and describing data by applying statistical techniques to the data (Neuman, 2011). For this study, quantitative data analysis was carried out. The quantitative analysis involves numbers and can use mathematical operations for the properties of the data (Walliman, 2017). Quantitative data analysis was carried out using the F-measure, Precision, Accuracy and Recall metrics.

3.6 Research Ethics

A research ethical clearance from the UNAM Research Ethical Clearance Committee (UREC) together with a research permission was obtained from UNAM Centre for Postgraduate Studies (CPGS). Permission to use data from real-world network environment available online was obtained through accepting the terms and conditions of using the datasets before downloading them. Permission was also obtained from UNAM Computer Centre to capture the university's network data. Confidentiality, integrity and anonymity of data was also preserved by not disclosing the obtained traffic data to other parties and by anonymising traffic traces.

3.7 Summary

This chapter presented the research methods used in this study. The chapter covered information about the quantitative research approach, the experimental research design, research instruments used in the study, research procedure, data analysis and

research ethics considered in this study. The next chapter will present the results and further discuss the results of the study.

Chapter 4: Results and Discussions

In this chapter, results and discussion for creating a traffic dataset from a real-world network (RO_1) and results obtained from training and testing the RNN models using several datasets (RO_2) are presented. Also, the performance evaluation results and discussion of the GRU model (RO_3) are presented.

4.1 Creation of a traffic dataset from a real-world network

A dataset was created using network traffic data from the University of Namibia's network. The network packets were captured and the attributes of the network packets include the packet's number, time, source, destination, protocol, length, info and class as shown in Table 7. A dataset from a real-world network was created as the first step since the design of the anomaly-based intrusion detection system starts with the collection of malicious and non-malicious behaviour for the network.

The network traffic dataset created from UNAM consisted of 55000 records of which each represents a network packet with 8 attributes. Through pre-processing, the features are converted to numerical data which can be used by the models. Out of the 55000 records, 126 records were regarded as malicious data.

Table 7. Network traffic attributes

Detail	Description
NO	ID of the packet
Time	Number of seconds elapsed
Source	IP address of the sender of the packet
Destination	IP address of the receiver of the packet
Protocol	Name of used protocol
Length	Length of the packet
Info	Additional information on the packet content.

For the records which were considered to be malicious or anomalous, there is no additional information attached to the data, in terms of the attack type. This is a limitation in the dataset as it lacks details about the anomalous traffic, however, it can be used effectively to detect intrusions. In addition to the lack of attack information in the dataset, the dataset created from real-world network traffic has an imbalanced proportion of non-malicious and malicious data. Imbalanced data in the dataset is caused by a high number of non-malicious behaviour traffic compared to malicious traffic. The state of real-word traffic dataset having more non-malicious data is a concern that Ring et al., (2019) pointed out as the limitation of using data from real-world networks. Although network traffic from real-world environments has an imbalanced proportion of non-malicious and malicious data, there is a need to create datasets from real-world network traffics to train models of IDSs to be used in real-world environments. Creating the dataset from real-world network traffic traces is

time-consuming as there are certain tasks which need to be done such as labelling and feature conversion which are labour-intensive. In addition, Tasks such as feature conversion need to be done for the data to be used by the models which work with numerical data.

4.2 Training and testing RNN models using several datasets

This section presents the results obtained from training and testing basic RNN, LSTM and GRU RNN models using real-world and synthetic datasets. The RNN models were trained and tested with UNAM dataset, Kyoto dataset and a combination of UNAM and Kyoto real-world datasets. The accuracy and recall performance metrics were used to evaluate the performance of the models.

As shown in Table 8, the basic RNN model obtained an accuracy score of 99.7%, with a recall score of 38.4% using UNAM dataset, 87.3% accuracy score with 20.7% recall score for Kyoto dataset and 52.2% accuracy score, with 27.7% recall score for Kyoto-UNAM dataset. For all the three datasets, the basic RNN scored an average accuracy score of 79.7% and an average recall score of 28.9%.

Table 8. Accuracy and recall results of basic RNN model

Dataset	**Accuracy**	**Recall**
UNAM	99.7%	38.4%
Kyoto University (2016) dataset	87.3%	20.7%
Kyoto-UNAM dataset	52.2%	27.7%
Accuracy / Recall Average	**79.7%**	**28.9%**

Table 9 shows that the GRU RNN model recorded an accuracy score of 99.3% with 88.0% recall score using the UNAM dataset, 97.2% accuracy score with 56.7% recall score for Kyoto-UNAM dataset and 87.3% accuracy score with 30.7% recall score for Kyoto dataset. For all the three datasets, the GRU RNN model obtained an average accuracy score of 94.6% and an average recall score of 58.5%.

Table 9. Accuracy and recall results of GRU RNN model

Dataset	**Accuracy**	**Recall**
UNAM	99.3%	88.1%
Kyoto University (2016) dataset	87.3%	30.7%
Kyoto-UNAM dataset	97.2%	56.7%
Accuracy / Recall Average	**94.6%**	**58.5%**

The LSTM model recorded an accuracy of 97.3% with 88.8% recall score using the UNAM dataset, 92.5% accuracy score with 37.7% recall score for Kyoto-UNAM dataset and 87.3% accuracy score with 22.3% recall score for Kyoto dataset as shown in Table 10. For all the three datasets, the LSTM model scored an average accuracy of 92.3% and an average recall score of 49.6%.

Table 10. Accuracy and recall results of LSTM model

Dataset	**Accuracy**	**Recall**
UNAM	97.3%	88.8%
Kyoto University (2016) dataset	87.3%	22.3%
Kyoto-UNAM dataset	92.5%	37.7%
Accuracy / Recall Average	**92.3%**	**49.6%**

The total average accuracy results of the different RNN models are calculated and shown in Table 11.

Table 11. Total accuracy and recall average of RNN models for real-world data

RNN Model	**Accuracy**	**Recall**
Basic RNN	80%	29%
GRU	95%	56%
LSTM	92%	50%

As shown in Table 11, the Basic RNN model scored an accuracy of 80% with a recall score of 29%, while the GRU model scored an accuracy of 95% with a recall score of 58% and LSTM scored an accuracy of 92% with a recall score of 50%.

The basic RNN, LSTM and GRU RNN models were also trained and tested with the NLS-KDD synthetic dataset. The accuracy and recall score of the models were calculated and the results are shown in Table 12.

Table 12. Accuracy and recall scores of RNN models using NLS-KDD data

RNN Model	Accuracy	Recall
Basic RNN	92.9%	93.1%
LSTM	94.5%	95.5%
GRU	97.0%	95.3%

The GRU model obtained the highest accuracy score of 97% with a recall score of 95.3%, followed by the LSTM model with an accuracy score of 94.5% with recall score of 95.5% and then the basic RNN model with an accuracy score of 92.9% and a recall score of 93.1%. The NLS-KDD synthetic dataset was used in the experiment in order to compare its performance to the real-world datasets. The results show that the models obtained high scores when a synthetic dataset is used compared to real-world dataset. Even though the scores obtained using synthetic are higher, the results obtained from using the real-world dataset shows how the models can perform in a real-world environment.

From the results obtained from training and testing the RNN models (basic RNN, LSTM and GRU), the GRU model obtained the highest average accuracy score of 95%, together with a high accuracy score and recall score of 99.3% and 88.1% respectively on the UNAM dataset. This shows that the models performed well in terms of accuracy with the UNAM real-world dataset which was created for this study. In addition, when all the three models were tested with synthetic data, the GRU model outperformed the LSTM and the basic RNN model. The GRU model is considered to have a simpler architecture and therefore, its simple architecture might have

contributed to the high accuracy score of the GRU model. The GRU model was considered as the model to be used in this study as it obtained accuracy and recall score which is higher compared to those of the basic RNN and the LSTM models. Since the GRU model obtained a high accuracy and recall score during training, there is a possibility that the GRU RNN –IDS could perform outstandingly with high rates of detecting intrusions. The study further compared the LSTM and GRU model to the basic RNN model. It was found that the basic RNN model obtained the lowest accuracy score and recall score on both real-world and synthetic datasets. The low accuracy and recall score of the basic RNN could be caused by the vanishing gradient problem experienced by the basic RNN architecture, which can lead to less accuracy compared to other models.

A comparative experiment by Xu et al (2018) got similar results to this study as their experiment showed that the GRU outperformed the LSTM and the GRU is more suitable for IDS than LSTM. The study by Xu et al (2018) made use of the KDD 99 and NSL-KDD synthetic datasets while our study made use of the real-world datasets to compare the performance of the models. This means that the GRU model has the capability of outperforming the LSTM, using real-world dataset or synthetic dataset, as shown by this study. The difference between their study and this study is that this study made use the k-fold cross-validation for training and testing the models. In this study, the use of K-Folds cross-validation method maximized the use of the limited available data for training and then testing the models using the different datasets. The K-Folds cross-validation method was found to be useful in this study for assessing the

performance of the model and using different data sets and making predictions on all of the datasets.

4.3 Performance evaluation of the GRU model

The performance of the GRU model was evaluated using the confusion matrix which gives a summary of the predicted results and the actual results in classifying intrusions using real-word dataset as shown in Table 13. For this study, a 2-class confusion matrix that describes the performance of a binary classification model is used.

Table 13. Confusion Matrix Results of GRU model using real-world dataset (UNIM)

		Actual Values	
		Negative	Positive
Predicted Values	Negative	TN = 54538	FP = 336
	Positive	FN = 15	TP = 111

Based on the results from Table 13, the confusion matrix results show that the GRU model made a total of 55000 predictions. A total of 54538 prediction cases were predicted not to be intrusions and they are not intrusions. This means that the model was correct to predict that no attack happened for those cases. A total of 111 prediction

cases were predicted to be intrusions and they are really intrusions. A total of 336 prediction cases were predicted as intrusions but they were actually not intrusions. The high number of false-positive is caused by the imbalanced number of negative and positive classes in the dataset were positive classes are few compared to the negative classes. Furthermore, from 126 intrusions in the dataset, only 15 prediction cases were predicted not to be intrusions, but they are actual intrusions. Additionally, the performance of the GRU model is also measured against Accuracy (AC), Precision (P), Recall (R) and F-measure (F), using the values from the confusion matrix. The results based on the evaluation metrics are shown in Table 14.

Table 14. Performance results of GRU model based on the evaluation metrics

AC	P	R	F
99.3%	24.8%	88.1%	38.7%

The results show that the model obtained an accuracy score of 99.3%, a precision score of 24.8%, recall score of 88.1% and F-measure score of 38.7%.

The performance of the GRU model indicates that it has high accuracy in binary classification for intrusion detection using real-world dataset, as it obtained an accuracy score of 99.3%. The accuracy metric is important when it comes to classification, as high accuracy is one of the desirable characteristics of a model (Choudhary & Swarup, 2009). The GRU model obtained a precision score of 24.8 % which is affected by the smaller number of positive data in the dataset and the number of false positives. The imbalanced nature of the dataset used caused the precision score

of the model to be low compared to other performance metrics. The false-positive can be made less by using sampling techniques, but the sampling techniques can introduce bias in the model results. On the other hand, when it comes to detecting intrusions, it is acceptable to have more false-positive than having more intrusions which are not detected by the model.

Based on the results obtained from evaluating the performance of the GRU model, the model scored an f-measure score of 38.7%. The F-measure score was affected by the precision score as the F-measure score is calculated by considering both the metrics of precision and recall score equally. In addition, the GRU model obtained a high recall score of 88.3%. The recall score of the model shows that the TPR of the model is high with 88.3% and the malicious traffic is correctly predicted. The models recall score also shows that the model can detect true positive events. Moreover, the high recall score obtained by the GRU model is important as it gives a picture of the true positive rate and also measures how sensitive the model is at detecting positive events which in this case are intrusions.

The results of the performance of the model show how the IDS could perform in a real-world environment when trained with real-world data which have a high possibility of imbalanced datasets and few features. Traffic datasets from real-world networks can present the existing real networks moderately, which will cause the IDS not to be biased in detecting anomalous behaviour on the real-world network, compared to using synthetic datasets. The results obtained when using the real-world dataset in training the model can allow further adjustments to the model for it to

improve its performance in the real-world environment. Furthermore, the performance of the model shows how the performance metrics such as precision are weakened by imbalance distributions of the classes in the dataset. By evaluating the performance of the GRU model, the third sub-objective (RO_3) has been achieved.

4.4 The Intrusion Detection System

After evaluating the performance of the GRU model, the model is then stored, and deployed using Tensorflow for it to be used as part of the RNN-IDS. TensorFlow-Serving is a high-performance serving system which is used to serve Tensorflow machine learning models (Olston et al., 2017), allowing a model to be deployed in production environments (Baylor et al., 2017). Model deployment is important as it allows the integration of the Machine Learning model with a production environment. In this study, TensorFlow Serving was used together with a Flask web server which was created using Flask version 1.0.2. Flask is a Python-based micro web framework which can be used to create a REST API that allows sending of data, and receive a prediction as a response (Flask, 2010). TensorFlow Serving together with the Flask web server makes up the model server.

The RNN-IDS works by saving network traffic from the real-world network, and the saved traffic is then sent to the model server. At the model server, the Flask APIs receive the traffic details through API calls, which then computes the predicted value based on the GRU model and then returns the prediction.

4.5 Summary

This chapter presented and further discussed the results of the study focusing on the objectives of the study. A traffic dataset from a real-world network was created, as a result, the first sub-objective (RO_1) was met. By training and testing the RNN models using real-world datasets and synthetic datasets, the second sub-objective (RO_2) was met. The performance of the GRU model was also evaluated which resulted in the third sub-objective (RO_3) being met. In this chapter, all the sub-objectives of the study are achieved, therefore it can be concluded that the objective of the (RO_m) is achieved as well. The next chapter will present the conclusion of the study.

Chapter 6: Conclusion

This study created a practical traffic dataset from a real-world network, carried out an experiment that compared RNN architectures (basic RNN, LSTM and GRU), offering insight into how the RNN architectures perform when trained and tested with real-world and synthetic datasets. Lastly, the study also evaluated the performance of the GRU RNN-IDS model using real-world dataset.

All the objectives of the study were met, and in relation to the objectives of the study, it is concluded that real-word datasets are important to train IDS model in order to present the environment in which the IDS will work in. This will allow the IDS not to be biased in the environment it will be used. Despite the effort to create a dataset from real-world network traffic, the dataset is imbalanced, with extremely more non-malicious behaviour traffic compared to anomalous traffic.

Furthermore, amongst the three RNN models, the GRU model showed that it is more accurate compared to other models as it obtained the best accuracy performance score on all datasets, and it obtained a good f-measure score compared to the other models.

The GRU model's performance evaluation shows that the model has good accuracy, precision, recall and f-measure for binary classification on the imbalanced dataset, even though there are cases of false-positive and negative results. Moreover, the results from the performance evaluation of the model using the chosen metrics give an overview of how it can perform in an environment with a possibility of imbalanced classes.

The contribution of this research to the body of knowledge is in the perspective of using real-world dataset for intrusion detection system. The theoreticaland practical contributions of this research includes a comparison of the RNN architectures (basic RNN, LSTM and GRU), offering insight into how the RNN architectures perform when trained and tested with real-world and synthetic datasets, and the evaluation of the performance of the GRU RNN-IDS model using real-world dataset.

Chapter 7: Recommendations

This study proposes the following recommendations, as emanated from the findings:

- There is a need to find methods to solve the imbalance of classes in real-world datasets, without turning the data into synthetic data.
- The models for the IDS need to be trained with data from real-world environments similar to which they will be used in and therefore, it is needed and important to consider the environment in order to choose the best model to be used in the environment.

References

Agarap, A. F. M. (2018). A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data. *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, 26–30. ACM.

Agrawal, G., Soni, S. K., & Agrawal, C. (2017). A Surveyon Attacks and Approaches of Intrusion Detection Systems. *International Journal of Advanced Research in Computer Science*, *8*(8), 499–504.

Ahmadzadegan, M. H., Khorshidvand, A. A., & Valian, M. G. (2015). Low-rate false alarm intrustion detection system with genetic algorithm approach. *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 1045–1048. https://doi.org/10.1109/KBEI.2015.7436188

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19–31.

Akosa, J. (2017). Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data. *In Proceedings of the SAS Global Forum*, 2–5.

AliShah, A., Sikander Hayat Khiyal, M., & Daud Awan, M. (2015). Analysis of Machine Learning Techniques for Intrusion Detection System: A Review. *International Journal of Computer Applications*, *119*(3), 19–29. https://doi.org/10.5120/21047-3678

Althubiti, S., Nick, W., Mason, J., Yuan, X., & Esterline, A. (2018). Applying Long Short-Term Memory Recurrent Neural Network for Intrusion Detection. *SoutheastCon 2018*, 1–5. IEEE.

Armah, G. K., Luo, G., & Qin, K. (2014). A Deep Analysis of the Precision Formula for Imbalanced Class Distribution. *International Journal of Machine Learning and Computing*, *4*(5), 417–422. https://doi.org/10.7763/IJMLC.2014.V4.447

Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., … Koc, L. (2017). Tfx: A tensorflow-based production-scale machine learning platform. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1387–1395. ACM.

Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, *5*(Sep), 1089–1105.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2015). Towards Generating Real-life Datasets for Network Intrusion Detection. *IJ Network Security*, *17*(6), 683–701.

Cermak, M., Jirsik, T., Velan, P., Komarkova, J., Spacek, S., Drasar, M., & Plesnik, T. (2018). Towards Provable Network Traffic Measurement and Analysis via Semi-Labeled Trace Datasets. *2018 Network Traffic Measurement and Analysis Conference (TMA)*, 1–8. IEEE.

Chetty, P. (2016). *Importance of research approach in a research.* Retrieved February 20, 2019, from https://www.projectguru.in/publications/selecting-research-approach-business-studies/

Chollet, F. (2018). *Deep Learning with Python*. United States of America: Manning Publications Co.

Choudhary, A. K., & Swarup, A. (2009). Neural network approach for intrusion detection. *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, 1297–1301. ACM.

Chowdhury, M. N., Ferens, K., & Ferens, M. (2016). Network intrusion detection using machine learning. *Proceedings of the International Conference on Security and Management (SAM)*, 30. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Deloche, F. (2017a). *File: Gated Recurrent Unit.svg*. Retrieved August 15, 2019 from https://en.wikipedia.org/wiki/File:Gated_Recurrent_Unit.svg

Deloche, F. (2017b). *File: Long Short-Term Memory.svg*. Retrieved August 15, 2019 fromhttps://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

Devarakonda, N., Pamidi, S., Kumari, V. V., & Govardhan, A. (2012). Intrusion Detection System using Bayesian Network and Hidden Markov Model. *Procedia Technology*, *4*, 506–514. https://doi.org/10.1016/j.protcy.2012.05.081

Dias, L. P., Cerqueira, J. J. F., Assis, K. D. R., & Almeida, R. C. (2017). Using artificial neural network in intrusion detection systems to computer networks. *2017 9th Computer Science and Electronic Engineering (CEEC)*, 145–150. IEEE.

Elsherif, A. (2018). Automatic Intrusion Detection System Using Deep Recurrent Neural Network Paradigm. *Journal of Information Security and Cybercrimes Research (JISCR), 1*(1), 28 - 41.

Eyisi, D. (2016). *The Usefulness of Qualitative and Quantitative Approaches and Methods in Researching Problem-Solving Ability in Science Education Curriculum. 7*(15), 91–100.

Flask. (2010). *Flask's documentation website.* Retrieved March 24, 2019, from http://flask.palletsprojects.com/en/1.1.x/

Haider, W., Hu, J., Slay, J., Turnbull, B. P., & Xie, Y. (2017). Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications, 87*, 185–192.

Hamid, Y., Sugumaran, M., & Journaux, L. (2016). Machine learning techniques for intrusion detection: A comparative analysis. *Proceedings of the International Conference on Informatics and Analytics*, 53. ACM.

Hunter, J., D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95.

Ingre, B., & Yadav, A. (2015). Performance Analysis of NSL-KDD dataset using ANN. *2015 International Conference on Signal Processing and Communication Engineering Systems*, 92–96. IEEE.

Jha, J., & Ragha, L. (2013). Intrusion detection system using support vector machine. *International Journal of Applied Information Systems (HAIS)-ISSN*, 2249–0868.

Jyothsna, V., & Prasad, K. M. (2019). Anomaly-Based Intrusion Detection System. In *Computer and Network Security*. IntechOpen.

Kadam, P. U., & Deshmukh, M. (2014). Various Approaches for Intrusion Detection System: An Overview. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(11), 6894 - 6902.

KDD Cup (1999). *KDD Cup 1999 Data*. Retrieved March 1, 2019, from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Keras. (2019). *Keras: The Python Deep Learning library*. Retrieved February 25, 2019, from https://keras.io

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, *2*(1), 2–20.

Kim, K., Aminanto, M. E., & Tanuwidjaja, H. C. (2018). *Network Intrusion Detection Using Deep Learning: A Feature Learning Approach*. Springer.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Kumar, G. (2014). Evaluation metrics for intrusion detection systems-a study. *Evaluation*, *2*(11), 11- 17.

Le, T.-T.-H., Kim, Y., & Kim, H. (2019). Network Intrusion Detection Based on Novel Feature Selection Model and Various Recurrent Neural Networks. *Applied Sciences*, *9*(7), 1 - 29.

Li, Z., Batta, P., & Trajkovic, L. (2018). Comparison of Machine Learning Algorithms for Detection... - Google Scholar. *IEEE*, 4248–4253.

Lin, W.-H., Lin, H.-C., Wang, P., Wu, B.-H., & Tsai, J.-Y. (2018). Using convolutional neural networks to network intrusion detection for cyber threats. *2018 IEEE International Conference on Applied System Invention (ICASI)*, 1107–1110. IEEE.

Liu, H., & Lang, B. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Applied Sciences*, *9*(20), 4396.

Liu, Y., Liu, S., & Zhao, X. (2017). Intrusion detection algorithm based on convolutional neural network. *2017 4th International Conference on Engineering Technology and Application (ICETA 2017)*.

Lotfallahtabrizi, P., & Morgan, Y. (2018). A novel host intrusion detection system using neural network. *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*, 124–130. https://doi.org/10.1109/CCWC.2018.8301663

McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc.

Mohammadpour, L., Ling, T. C., Liew, C. S., & Chong, C. Y. (2018). A Convolutional Neural Network for Network Intrusion Detection System. *Proceedings of the Asia-Pacific Advanced Network*, *46*, 50–55.

Mylavarapu, G., Thomas, J., & Kumar, A. (2015). Real-time Hybrid Intrusion Detection System using Apache Storm. *IEEE*. https://doi.org/10.1109/HPCC-CSS-ICESS.2015.241

Neuman, L. (2011). *Social Research Methods: Qualitative and Quantitative Approaches*. Pearson.

Olston, C., Fiedel, N., Gorovoy, K., Harmsen, J., Lao, L., Li, F., … Soyke, J. (2017). *TensorFlow-Serving: Flexible, High-Performance ML Serving. ArXiv:1712.06139 [Cs]*. Retrieved February 18, 2019, from http://arxiv.org/abs/1712.06139

Pandas. (2019). *Python Data Analysis Library*. Retrieved February 20, 2019, from https://pandas.pydata.org/

Ponkarthika, M., & Saraswathy, V. R. (2018). Network Intrusion Detection Using Deep Neural Networks. *Asian Journal of Applied Science and Technology*, *2*(2), 665–673.

Prajyot, A., & Kalavadekar, P. (2018). Review on Intrusion Detection System using Recurrent Neural Network with Deep Learning. *International Research Journal of Engineering and Technology*, *05*(10), 1385-1388.

Protic, D. D. (2018). Review of KDD Cup'99, NSL-KDD and Kyoto 2006+ datasets. *Vojnotehnicki Glasnik*, *66*(3), 580–596.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A Survey of Network-based Intrusion Detection Data Sets. *Computers & Security, 8,* 147-167.

Sarmah, A. (2001). *Intrusion Detection Systems: Definition, Need and Challenges*. Retrieved March 18, 2019, from

https://www.sans.org/readingroom/whitepapers/detection/intrusion-detection-systems-definition-challenges-343

Satam, P. (2015). *An Anomaly Behavior Analysis Intrusion Detection System for Wireless Networks* (The University of Arizona). Retrieved May 10, 2019, from https://pdfs.semanticscholar.org/4e10/53d184865f74a56d676429f3bf18cf9b72e2.pdf

Saxena, P. (2017). Analysis of Network Traffic by Using Packet Sniffing Tool: Wireshark. *International Journal of Advance Research, Ideas and Innovations in Technology, 3*(6), 804-808.

Scikit-learn. (2019). *Scikit-learn Machine Learning in Python*. Retrieved March 10, 2019, from https://scikit-learn.org/stable/

Shah, A. A., Hayat, M. S., & Awan, M. D. (2015). *Analysis of Machine Learning Techniques for Intrusion Detection System: A Review*. Infinite Study.

Sharma, V., & Nema, A. (2013). Innovative Genetic Approach for Intrusion Detection by Using Decision Tree. *2013 International Conference on Communication Systems and Network Technologies*, 418–422. https://doi.org/10.1109/CSNT.2013.93

Simon, M. K., & Goes, J. (2013). *Scope, limitations, and delimitations*. Retrieved February 18, 2019, from http://www.dissertationrecipes.com/wp-content/uploads/2011/04/limitationscopedelimitation1.pdf

Singh, J., & Nene, M. J. (2013b). A survey on machine learning techniques for intrusion detection systems. *International Journal of Advanced Research in Computer and Communication Engineering*, *2*(11), 4349–4355.

Takakura. (2015). *Traffic Data from Kyoto University's Honeypots*. Retrieved March 15, 2019, from http://www.takakura.com/Kyoto_data/

TensorFlow. (2019). *An end-to-end open source machine learning platform*. Retrieved February 08, 2019, from TensorFlow website: https://www.tensorflow.org

Tiwari, P. (2018). *My Tech World*. Retrieved May 22, 2019, from https://tekworld.org/2018/12/28/day-47-100-days-mlcode-recurrent-neural-network/#page-content

Tompkins, J. (2019). *Disinformation Detection: A review of linguistic feature selection and classification models in news veracity assessments*. Retrieved September 19, 2019, from https://arxiv.org/pdf/1910.12073.pdf

UNB. (2015). *NSL-KDD dataset*. Retrieved February 08, 2019, from https://iscxdownloads.cs.unb.ca/iscxdownloads/NSL-KDD/#NSL-KDD

Vani, R. (2017). Towards Efficient Intrusion Detection using Deep Learning Techniques: A Review. *Int*ernational Journal of Advanced Research in Computer and Communication Engineering, *6(10),*375 -384.

Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Applying convolutional neural network for network intrusion detection. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1222–1228. https://doi.org/10.1109/ICACCI.2017.8126009

Walliman, N. (2017). *Research methods: The basics*. Routledge.

Wang, J., Yang, Q., & Ren, D. (2009). An intrusion detection algorithm based on decision tree technology. In *2009 Asia-Pacific Conference on Information Processing*, 333-335. IEEE.

Willems, K. (2017). *Python Exploratory Data Analysis Tutorial.* Retrieved August 15, 2019, from https://www.datacamp.com/community/tutorials/exploratory-data-analysis-python

Xu, C., Shen, J., Du, X., & Zhang, F. (2018). An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units. *IEEE Access*, *6*, 48697–48707.

Yadav, S., & Shukla, S. (2016). Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, 78–83.

Yeo, L. H., Che, X., & Lakkaraju, S. (2017). Understanding Modern Intrusion Detection Systems: A Survey. *ArXiv Preprint ArXiv:1708.07174*.

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, *5*, 21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418

Zamani, M., & Movahedi, M. (2013). Machine learning techniques for intrusion detection. *ArXiv Preprint ArXiv:1312.2177*.

Appendix 1: Research Permission Letter

University of Namibia, Private Bag 13301, Windhoek, Namibia
340 Mandume Ndemufayo Avenue, Pioneers Park
☎ +264 61 206 3111; URL: http://www.unam.edu.na

UNAM
UNIVERSITY OF NAMIBIA

02 April 2019

Ms. Hilma Aludhilu (MSC: IT candidate)
Department of Information Technology
School of Computing
University of Namibia

RE: APPROVAL FOR PERMISSION TO CONDUCT RESEARCH TITLED "AN INTRUSION DETECTION SYSTEM USING RECURRENT NEURAL NETWORK WITH REAL-WORLD DATASET."

Reference is made to the above heading.

I am pleased to inform you that approval has been granted to Ms. Hilma Aludhilu to conduct research on our network and interview staff from Computer Centre. Further approval to carry out experiments under supervision of Computer Centre staff necessary for the study is hereby also given, as the findings of study will be useful to assist us improve on information security at the University.

UNIVERSITY
2019 -04- 02

Please feel free to contact me for more details, on 061 206 3031 or email me on hamata@unam.na

Sincerely,

Tuyoloni Hamata
Director: Information & Communication Technology Services