8-24-2018 9:30 AM

Recurrent Neural Network Architectures Toward Intrusion Detection

Wafaa Anani, *The University of Western Ontario*

Supervisor: Jagath Samarabandu, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

Recurrent Neural Networks (RNN) show a remarkable result in sequence learning, particularly in architectures with gated unit structures such as Long Short-term Memory (LSTM). In recent years, several permutations of LSTM architecture have been proposed mainly to overcome the computational complexity of LSTM. In this dissertation, a novel study is presented that will empirically investigate and evaluate LSTM architecture variants such as Gated Recurrent Unit (GRU), Bi-Directional LSTM, and Dynamic-RNN for LSTM and GRU specifically on detecting network intrusions. The investigation is designed to identify the learning time required for each architecture algorithm and to measure the intrusion prediction accuracy. RNN was evaluated on the DARPA/KDD Cup'99 intrusion detection dataset for each architecture. Feature selection mechanisms were also implemented to help in identifying and removing non-essential variables from data that do not affect the accuracy of the prediction models, in this case Principal Component Analysis (PCA) and the RandomForest (RF) algorithm. The results showed that RF captured more significant features over PCA when the accuracy for RF 97.86% for LSTM and 96.59% for GRU, were PCA 64.34% for LSTM and 67.97% for GRU. In terms of RNN architectures, prediction accuracy of each variant exhibited improvement at specific parameters, yet with a large dataset and a suitable time training, the standard vanilla LSTM tended to lead among all other RNN architectures which scored 99.48%. Although Dynamic RNN's offered better performance with accuracy, Dynamic-RNN GRU scored 99.34%, however they tended to take a longer time to be trained with high training cycles, Dynamic-RNN LSTM needs 25284.03 seconds at 1000 training cycle. GRU architecture had one variant introduced to reduce LSTM complexity, which developed with fewer parameters resulting in a faster-trained model compared to LSTM needs 1903.09 seconds when LSTM required 2354.93 seconds for the same training cycle. It also showed equivalent performance with respect to the parameters such as hidden layers and time-step. BLSTM offered impressive training time as 190 seconds at 100 training cycle, though the accuracy was below that of the other RNN architectures which didn't exceed 90%.

Keywords: Recurrent Neural Networks, Gated Recurrent Unit, Long Short-term Memory, Skip-LSTM, Bi-Directional LSTM, Dynamic-RNN, Intrusion Detection, Deep Learning.

Acknowlegements

Foremost, I would like to express my sincere gratitude to my advisor Dr. Jagath Samarabandu of the Electrical and Computer Engineering department at Western University, for the continuous support of my MESc study and research, as well as his patience, motivation, enthusiasm, and immense knowledge. The door to Dr. Samarabandu's office was always open whenever I was stuck on an issue or had a question to ask. He consistently directed me in the right path whenever needed. His guidance has assisted me in writing of this thesis.

I would also like to sincerely thank my labmates, Gobi and Nadun, for their companionship during this time and their feedback during our invaluable lab meetings.

I must express my gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. And finally to my sister Lina, who experienced all the ups and downs that I faced in my research, and kept me motivated.

Contents

List of Figures

List of Tables

List of Abbreviations

BLSTM Bi-Directional LSTM

DRNN Dynamic-RNN

GRU Gated Recurrent Unit

IDS Intrusion Detection Systems

LSTM Long Short-term Memory

PCA Principle Component Analysis

RF RandomForest

RNN Recurrent Neural Network

Chapter 1

Introduction

1.1 Overview

Intrusion detection is a key research area in network security. A common approach for intrusion detection is detecting anomalies in network traffic, however, network threats are evolving at an unprecedented rate. The difference between the evolution of threats and the current detection response time of a network leave the system vulnerable to attacks [1]. Over the years, a number of machine learning techniques have been developed to detect network intrusions using packet prediction [2]. Recurrent Neural Network (RNN) is the most popular method of performing classification and other analysis on sequences of data. A subset network of RNN is Long Short-term Memory (LSTM), introduced by Hochreiter and Schmidhuber (1997) [3]. LSTM is a key algorithm in regards to the implementation of machine learning tasks that involve sequential data. Successful deployment of LSTM has led the industry to heavily invest in implementing the algorithm in a wider range of applications. These applications include voice recognition [4], [5], handwriting recognition [6], machine translation and social media filtering, thus making LSTM a natural candidate for Intrusion Detection Systems (IDS). Yet, this

algorithm incurs high computational costs when it is deployed on a large scale, in both time and memory complexity [7], [8]. To overcome this challenge, several variations of the algorithm have been proposed. These variations include Gated Recurrent Unit (GRU), Bi-Directional long short-term memory (BLSTM), Dynamic-RNN for LSTM and GRU, and Skip-RNN. This thesis presents a novel empirical study investigating and implementing the variants of LSTM architectures for intrusion detection based on predicting packet sequences. The implementation of each architecture was evaluated in terms of training time, prediction accuracy (normal or intrusion), the sensitivity of parameters, as well as several performance metrics including precision, recall, and false alarm rate. Experiments were conducted on the full KDD Cup'99 - intrusion detection dataset [9], these algorithms were evaluated on the entire data set, rather than on the most commonly used KDD 10% dataset used in the majority of intrusion detection literature.

1.2 Motivation

The Long Short-term Memory (LSTM) is a subset network of the RNN, an architecture that excels at storing sequential short-term memories and retrieving them many time-steps later. For example, RNN has the capability to learn from previous time-steps of the input data. The data at each time-step is processed and stored and given as input to the next time-step. The algorithm at the next time step utilizes the previous data stored to process the information. Such architecture, with a robust computation power would be suitable for security applications, in particular dealing with streaming data such as network sequence packets. The security domain is always researching to catch up with the evolution of intrusion. The new field of RNN in intrusion detection is still in the initial stages of research and has an immense potential for adaptation of these gated algorithms to learn insights much faster and provide intrusion detection close to real-time. Though the neural network structures are complex, with the right

set of parameters they can be tuned to obtain light-weight functionality. This served as the motivation to explore gated RNNs and focus on the comparison between Long Short-term Memory (LSTM) and other variant versions of it such as GRU, BLSTM, Dynamic-RNN for LSTM/GRU and Skip-RNN.

1.3 Problem Statement

The main goal of this dissertation is to explore and analyze various RNN architectures, tune each architecture with a different set of parameters such as hidden layers, time-steps, training cycle and learning rate with the goal of identifying the best parameters to achieve a shorter time in training each algorithm and high accuracy in predicting whether a network stream packet is an intrusion or not.

I am looking to answer the following questions:

- What is the best architecture for the intrusion detection domain?
- What is the set of parameters that helps to achieve high accuracy and less time in training?
- What is the impact of feature selection on each algorithm?

1.4 Purpose, Scope, and Contribution

The purpose of this research is to evaluate different RNN algorithms on an intrusion detection dataset. The best known algorithms were identified, such as LSTM, GRU, BLSTM, and Dynamic-RNN LSTM/GRU. This research has an immense potential to open doors for improving the intrusion detection applications domain as it offers a better detection rate in catching

any attack before network security is compromised. The scope of this project is to find the most suitable architecture that should fulfill the research purpose by evaluating and analyzing the performance of the selected algorithms in terms of prediction accuracy and time required for each algorithm to be trained on an intrusion detection dataset. This study didn't measure detection time which is the time elapsed between the initial breach of a network by an attacker and the discovery of that breach.

The main contributions of this thesis are summarized below:

- Implemented, evaluated and compared the different RNN algorithms (LSTM, GRU, BLSTM, and DRNN LSTM/GRU) in terms of training time and prediction accuracy.
- Identified the sensitivity of each algorithm with respect to its individual parameters, such as learning rate, hidden layers, and training cycles, then measured the prediction accuracy
- Introduced for the first time DRNN LSTM/GRU to the intrusion detection domain.
- Calculated the performance metrics including precision, recall, and false alarm rate for each algorithm.
- Two algorithms for classification mechanism called RandomForest (RF) and Principal Component Analysis (PCA) are presented for feature selection in the domain of intrusion detection. A comparison between the two algorithms was conducted to find the most suitable algorithm to represent the data with the best performance in terms of prediction accuracy.
- Overall results were presented illustrating the best-case scenario obtained for each algorithm to be employed in the intrusion detection domain.
- The proposed LSTM optimized model scored 99.43%. Where as 19,593 more attacks out of 3,925,650 have been correctly detected when compared with LSTM models.

1.5 Research Methodology

Domain issues and challenges were identified with regards to intrusion detection through literature reviews.The most often used algorithms and experiments conducted were identified as well as the accuracy rate for each. As a result, it became evident that RNN architectures are new techniques in the intrusion detection domain. Proof of the concept of using LSTM and GRU algorithms in the field of intrusion detection is scarce due to lack of intensive experiments. None of the literature consulted demonstrated the best architecture or were compared among the algorithms in terms of their parameters to achieve the best performance with high accuracy. There are some challenges facing IDS which make it difficult to achieve that goal. Classification of data and labeling of unlabeled data seems to be a challenging task, as the current high volume of network traffic increases the number of attacks.

A module for each selected architecture was therefore developed. Feature selection was then implemented to ensure the best representation of all the data and better represent the underlying problem to each prediction model, resulting in improved model accuracy on unseen data. In this research, two phases were demonstrated for the experiment. One can be described as the features selection phase, using the two different selection mechanisms, Principal Component Analysis (PCA) and RandomForest (RF). RF learns from inputs and improves performance over time. With regards to intrusion detection, the aim is for the algorithm to learn over time which are the best features from the network features. PCA selects a new feature set to reduce redundancy of the features and improves performance [10], [11]. It had to be decided which algorithm would offer better performance for the intrusion detection domain, and to be implemented in IDS in real-time traffic. The second phase is to attempt to evaluate the selected algorithms on a full KDD Cup'99 dataset which is a commonly used intrusion detection dataset. A baseline was created with initial values for each parameter, based on literature review. Different values were used with each run to fine tune the parameters and to identify the suitable ones that showed best prediction accuracy. Research methodology illustrated in

Figure 1.1.

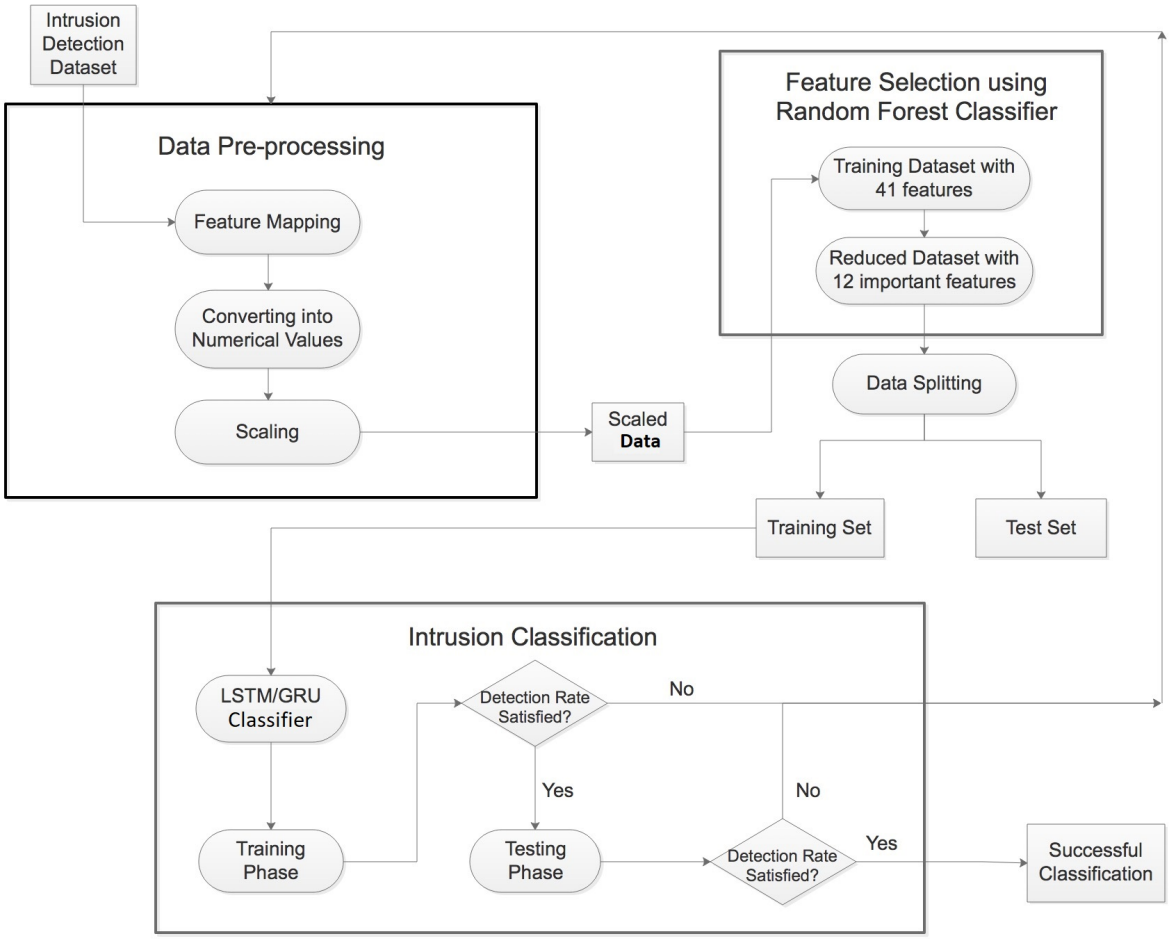

Figure 1.1: Research Methodology.

1.6 Thesis Organization

Chapter 2 provides a literature review related to security, in particular using a machine learning mechanism for intrusion detection. The main focus is on RNNs architecture and how far other researchers have advanced in the field. Chapter 3 demonstrates each algorithm architecture and its equations, along with all related topics including intrusion detection, parameter definitions, and evaluation matrices used in this research. In Chapter 4, experimental results are

demonstrated highlighting the dataset description, preprocessing and the selection of features using RF and PCA. In Chapter 5 the result of the experiment is reported, as well as a discussion of the performance of each algorithm with an overall analysis. This thesis is concluded in Chapter 6 by summarizing the work carried out, the contribution made, and the conclusions from the results obtained. Further research areas are also outlined in light of the needs of securing the network under the IoT applications and investigating deep learning and having a hybrid framework with different layers of different architecture.

Chapter 2

Literature Review

This section focuses on highlighting the related work for anomaly detection techniques in combination with machine learning algorithms. In particular, LSTM architectures, as it is considered to be a special kind of RNN that has the form of a chain of repeating models of a neural network. These repeating models, called "memory cells" as illustrated in Figure 2.1, contain four "gates" that can handle storing, finding long-range dependencies, and determining what information to keep or forget [12]. LSTM architecture is widely used in sequential data problems, especially ones related to natural language [5], [13], [14]. LSTM and its variances are being used in many studies in the field of intrusion detection. Tuor *et al.* [15] presented an online deep learning method for intrusion detection due to it's excellent ability to learn patterns, where they employed deep neural network autoencoders for unsupervised network anomaly detection using time aggregated statistics as features. Streaming scenarios were developed that utilize user logs to detect insider threats. Their evaluation conducted on the CERT dataset indicated that LSTM outperformed other anomaly detection techniques including Isolation Forest, SVMs, and PCA.

Yunsheng Fu. *et al.* [16] proposed an intelligent attack detection method in social net-

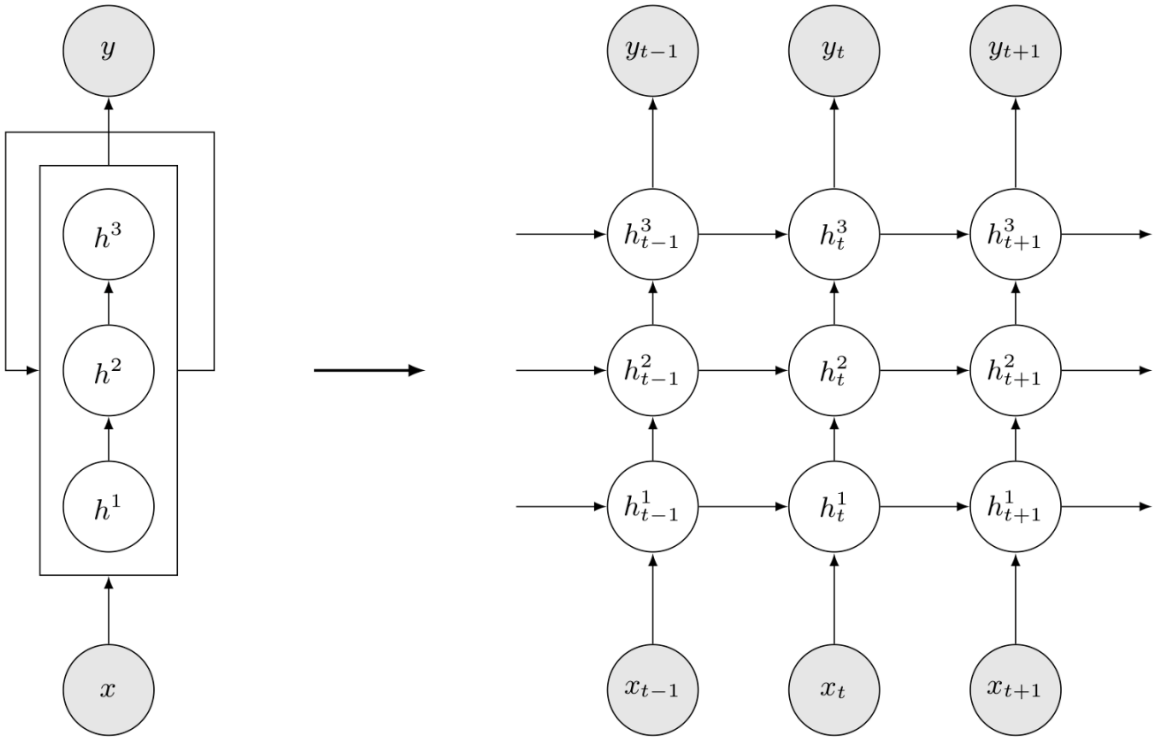

Figure 2.1: LSTM "Memory Cell".

works based on LSTM, with the purpose of achieving a high detection rate. They used the NSL-KDD dataset to evaluate the performance of their proposed method. Their experiment consisted of data preprocessing, feature abstraction, training and detection. LSTM were used during the training stage to classify whether the traffic was an attack or normal traffic. Experimental results demonstrated that their proposed intelligent attack detection method achieved state-of-the-art performance and is much faster than most post-processing algorithms. They compared their result with well known classifiers such as Bayesian, SVM, KNN, RBNN, PNN and GRNN. The intelligent attack detection method scored 98.85%. Within that context LSTM was employed in another well-known intrusion detection dataset, the full KDD Cup'99, as it was also proven that LSTM outperformed other anomaly detection algorithms as Al-kasassbeh *et al.* [17] showed that RF with higher accuracy at 93.78% and Meena *et al.* [18] showed accuracy for J48 and Naïve Bayes at 99.49% and 92.72%, respectively. Yunsheng and his team proved that LSTM outperformed the other well-known classifiers with a satisfactory result at

98.85%. 99.43% was managed to be scored in LSTM and 99.34% for DRNN GRU at 100 training cycles, and GRU at 1000 training cycles.

Kim *et al.* [19] showed effective results using LSTM with different values for learning rate and hidden layer size with high detection rate and accuracy. They applied their experiment on 10% of the KDD Cup '99 training dataset and 10% of the KDD Cup'99 testing dataset. Depending on the tuning of the algorithm's parameter values, the performance of the algorithm changed. Thus, the learning rate and numbers of hidden layers had a great impact on the performance. Authors found that the detection rate and false alarm rate showed improvement precisely when the learning rate parameter was set to 0.01 and hidden layers size set to 80, at which they register detection rate (0.877) and false alarm rate (0.133). According to their experiments, the average detection rate was 98.8% among the total attacks, and the average false alarm rate was 10%. Based on their experiments, most attacks like DoS and normal instances were detected. However, U2R instances were never detected since there weren't enough instances, with only 30 examples. Staudemeyer has trained LSTM on various network topologies to identify the suitable LSTM network parameters and structure [20]. Their results showed that the LSTM classifier has managed to detect "DoS" attacks and network "probes" despite the distant time series of events between each attack. LSTM outperforms the winning entries of the KDD Cup'99 challenge as LSTM can look back in time and correlate consecutive connection samples. Other researchers addressed the computational complexity by modifying LSTM implementation. Most literature used the KDD Cup'99 10% dataset to prove the efficiency of vanilla LSTM in predicting traffic anomalies and selecting optimal parameters to enhance its performance [18] [19]. For this research, the full KDD Cup'99 dataset was used, as LSTM showed a higher accuracy at 99.43% for the same learning rate of 0.01. Kim *et al.* had 80 hidden layers [19], in this experiment it was found that the best detection could be achieved at 50 hidden layers with less training time.

Miao *et al.* [7] have simplified LSTM based on their analysis of activation function of the

gates, and by identifying the redundancy in LSTM structure. They proposed two simplifications: (1) deriving input gates from forget gates, and (2) removing recurrent inputs from output gates. Lyu *et al.* [12] also proposed another simplification for LSTM. Moreover, Gers and Schmidhuber [21] introduced "peephole" connections to improve the ability of LSTM to learn precise timings and counting of the internal states. Peephole connections allow the gates to not only depend on the previously hidden state *(St-1)*, but also on the previous internal state *(Ct-1)*, adding additional terms in the gate equations. Greff *et al.* showed in their LSTM variants analysis that "peephole connections" did not resolve the problems that LSTM were tested on [8]. Based on that survey, any "peephole connection" was avoided within the implementation by focusing on original architecture for LSTM. The simplification Miao introduces did not really improve the complexity of the LSTM much [7], which is why GRU was the focus and considered it as LSTM with only two gates. It showed an improvement in the training time, yet didnot outperform LSTM until training cycles were increased.

Greg *et al.* [22] introduced GRU as a light version of LSTM. The architecture addressed the complexity of LSTM by eliminating the "output gate", which writes the contents from its memory cell to the more substantial net at each time step. Many studies implemented GRU to evaluate its performance in intrusion detection, as it is well-suited to classify, process, and predict time series. One investigation by Athiwaratkun and Stokes presented a new, two-stage malware classification model which utilizes a language model to generate the features. Then one single stage, character-level for malware classification. Their new malware language-model-based utilized LSTM or GRU to construct the features [23]. BLSTM, another form of LSTM architecture, was used in acoustic modeling in speech recognition [24].

Ahmed E. [25] addresses one of the intrusion detection system challenges, which is to achieve a low false alarm rate with new unseen threats. The author built a model using different RNN models to identify seen and unseen threats. Bi-Directional RNN, LSTM, BLSTM, are used to detect anomalies in sequence. He tested the models on NSL-KDD dataset. The results

show that BLSTM showed superiority over the other RNN models. He ties that to the fact that RNN has the ability to define normal behavior from large datasets and can be used to detect a new unseen threat.

Ali H. *et al.* [26] utilized LSTM for computer network intrusion detection with their proposed autoencoder framework for both fixed and variable length data sequence. They used LSTM encoders such as GRU, and BLSTM through a comprehensive set of experiments. They developed an online sequential unsupervised dataset for network intrusion detection using LSTM-autoencoders. Their experiment carried 5-folds cross validation that validate the performance of their framework using the ISCX IDS 2012 dataset. The experiment carried different autoencoders such as LSTM-Autoencoder with Last pooling, LSTM-Autoencoder with Max pooling, LSTM-Autoencoder with mean pooling and Deep Auto LSTM. They demonstrate that LSTM-Autoencoder with Max pooling showed the best f1-score.

GRU and BLSTM were both implemented in this research focusing on what each architecture offers in terms of performance within the intrusion detection dataset. As shown in Greg, Athiwaratkun, Ahmed E. and Ali literatures, their focus was on proving that RNN architecture has great potential in anomaly detection [22] [23] [25] [26]. In this experiment, a more in-depth analysis is offered on what each architecture could offer with the right set of parameters. GRU accelerated in showing strong results which could compete with LSTM. However, BLSTM showed superiority in training time, but didn't score high on accuracy, as Ahmed E. stated [25].

Thi-Thu-Huong Le *et al.* [27] built a classifier of IDS using LSTM with six optimizers: RMSprop, Adagrad, Adadelta, Adam, Adamax, and Nadam. They evaluated the performance of each optimizer using the KDD Cup'99 dataset on each attack type as follows: DoS, Probe, R2L, U2R and Normal. The main purpose of their experiment was to enhance the classification performance including accuracy, detection rate, and decreasing false alarm rate. Moreover, improving the classification result for each attack. Their experiment consists of two stages. The

first stage determined hyperparameter values. The second stage applied LSTM with the six optimizers. They concluded that the LSTM RNN model using the Nadam optimizer with learning rate 0.002 obtains the best results, and outperformed other classifiers with detection rate and FAR of 98.95% and 98.98%, respectively. Optimizers play a very crucial role to increasing the accuracy of the model. RMSProp, AdaDelta and Adam are very similar algorithms, and since Adam was found to slightly outperform RMSProp, Adam is generally chosen as the best overall choice. For this reason Adam Optimizer with LSTM was implemented.

Bontemps *et al.* [28] introduce a collective anomaly detection model using LSTM in real-time network traffic. LSTM trained with a normal time serious data without anomalies, rather than relying on the prediction errors in conjunction with detection rules to signal for anomalies. They demonstrate the efficient performance of the proposed model using the KDD Cup'99 dataset. The results showed the capability of the model to detect collective anomalies. However, they suggested that their model training data must be organized in a coherent manner to guarantee the stability of the system.

Recent work by Benjamin *et al.* [29] has demonstrated that LSTM RNN can be applied to the problem of anomaly detection in computer network flow data. They utilized a public dataset "ISCX IDS" for IDS taken from the University of New Brunswick's Canadian Institute for Cybersecurity (CIC) and the Information Security Centre of Excellence (ISCX). Their model consists of two stacked, bidirectional, LSTM layers, a single dense layer activation, and a single fully connected SoftMax output layer. Their model can identify anomalous network traffic. Observed anomalies were the focus for training the models for prediction attacks. The concept introduced by Bontemps [28] and Benjamin [29] could be a good starting point for future comparisons of this concept among all the RNN architecture from the point of having different layers within one framework, and trying to implement the unseen threats concept to observe which architecture would lead to a better performance.

Furthermore, SKIP-RNN is a newly proposed architecture by Campos *et al.* [30] in which

they extend the existing LSTM model by learning to skip state updates to reduce the number of sequential operations and the effective size in the computational graph. Their experiment was based on developing SKIP-LSTM and SKIP-GRU on the MNIST dataset (a large database of handwritten digits that is commonly used for training various image processing systems by Modified National Institute of Standards and Technology database). All parameters were trained using backpropagation. The results showed that Skip-RNN matched some cases and even outperformed the baseline models in other instances. There was an attempt to include this architecture into the research, by implementing it on the KDD Cup'99 dataset at different learning rates 0.01, 0.001, and 0.0001. Unfortunately the algorithm did not converge despite several attempts to train the model with different set values of its parameters.

In comparison to existing literature, this research offers insight into RNN architectures. Different feature selection techniques were compared, the best technique that fit intrusion detection was selected, and applied to a different machine learning model for intrusion detection. This resulted in presenting suitable parameter tuning to be changed while increasing the accuracy for this set of algorithm, which will be explained in detail in the results and analysis section. However, it is worth pointing out that the best accuracy was LSTM: 99.43% at 500 training cycle with 50 hidden layers. GRU accuracy was at 99.34% at 1000 training cycle. Dynamic-RNN LSTM was at 99.27% and Dynamic-RNN GRU at 99.34%, both at 100 training cycle.

Chapter 3

Background

This chapter presents information on the background for the research presented in this thesis. It introduces the concepts and technology relevant to the application of RNN algorithms in intrusion detection. These concepts include intrusion detection and machine learning. This is followed by a description of architectures of RNN, LSTM, GRU, BLSTM, and DRNN LSTM/GRU. Two feature selections algorithms, RF and PCA, are then explained. The final sections of the chapter explains the parameters definitions and the evaluation metrics used for this research.

3.1 Intrusion Detection and Machine Learning

Any misuse of a network that compromises its stability or the safety of its data is called a network intrusion attack [31]. Intrusion Detection (ID) is an essential component of network security [32] and has been defined as a technique that monitors system and network actions to identify abnormal and malicious activities including attack attempts in the network [33]. ID aims to identify and scan network activity and detect such intrusion attacks. ID's

biggest challenge is being able to identify those attacks efficiently, accurately and in a timely manner. Traditional systems were designed to find better-known attacks, however cannot determine unknown threats. Machine learning is the science of getting computers to act without being explicitly programmed. From an intrusion detection perspective, machine learning can be applied, data mining and pattern recognition algorithms to distinguish between normal and malicious traffic. A great deal of consideration has been given to machine learning techniques taking an immense part in many IDSs yet the research shows that using machine learning technique for detecting an attack may not perform identically in detecting another attack. The analysis of different machine learning algorithms has been completed in an evolutionary way [34], however the main goal of these machine learning techniques is to distinguish the intrusions for better preparation against future attacks. The simulated attacks fall in one of the following four categories:

- **Denial of Service Attack (DoS)**: is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.

- **User to Root Attack (U2R)**: is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.

- **Remote to Local Attack (R2L)**: occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.

- **Probing Attack**: is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

3.2 Recurrent Neural Network (RNN)

In this research RNN was selected due to its powerful features to learn from previously data, then adapt its response to predict. Furthermore, it combines two key features: 1) Distributed hidden state that allows them to store a lot of information about the past efficiently. 2) Non-linear dynamics that allow them to update their hidden state in complicated ways. RNN, an extension of a conventional feed forward neural network, is designed to recognize patterns in a data sequence. The RNNs are called recurrent because they perform the same task for every item of a sequence with the output being dependent on the previous computations [35]. The sequential information is preserved in the recurrent network's hidden state, which manages to span many time-steps as it cascades forward to affect the processing of each new input. It finds correlations between events separated by many moments, and these correlations are called "long-term dependencies" because an event downstream in time depends upon, and is a function of, one or more events that came before. One way to think about RNNs is to view them as a way to share weights over time, as illustrated in Figure 3.1.

To calculate RNN hidden state and output, equation 1 and 2 were used:

$$h_t = \sigma(Wh_{t-1} + Ux_t + b_t) \tag{3.1}$$

$$O_t = softmax(W^s h_t) \tag{3.2}$$

Where σ is a sigmoid function, x_t is an input vector at time t, h_t is a hidden state vector at time t, W is an input to hidden weight matrix, U is a hidden to hidden weight matrix, and b_t is a bias term.

Unfortunately, if RNN was implemented it would not give a satisfactory result. That is because the RNN model has a major drawback called the vanishing gradient problem. This means that the result won't be accurate, since at each time-step during training the same weights

are used to calculate the output O_t.

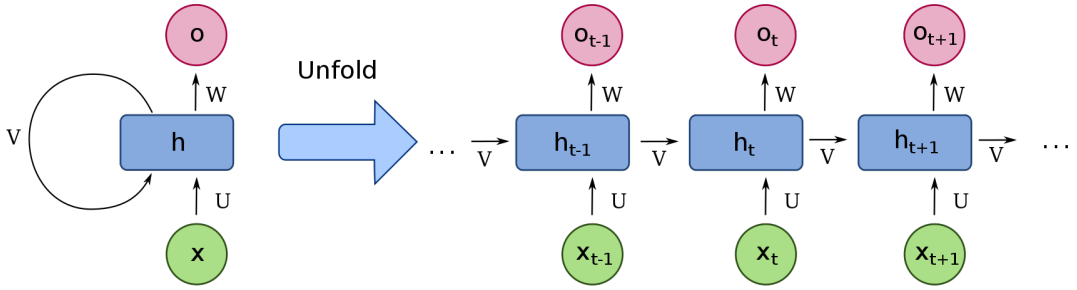

Figure 3.1: Simple RNN Architecture.

3.3 Long Short-term Memory (LSTM)

LSTM is a variation of recurrent network proposed as one of the machine learning techniques to solve many sequential data problems. LSTM helps to preserve the error that can be back-propagated through time and layer. At first, the LSTM cell is precisely introduced to reduce the multiplication of the gradient problem, as well as to make the RNN more useful for long-term memory tasks. LSTM architecture as illustrated in Figure 3.2 consists of four main components; the input gate (i), the forget gate (f), the output gate (o), and the memory cell (c). The cell makes decisions about what to store, read and write via gates that open or close, and each memory cell corresponds to a time-step. These gates pass the information based on a set of weights. Some of the weights, like input and hidden states, are adjusted during the learning process. Equations governing the operations of LSTM architecture are given below:

$$F_t = \sigma(W_F x_t + U_F h_{t-1} + b_F) \tag{3.3}$$

$$I_t = \sigma(W_I x_t + U_I h_{t-1} + b_I) \tag{3.4}$$

$$O_t = \sigma(W_O x_t + U_O h_{t-1} + b_O) \tag{3.5}$$

$$c_t = F_t \odot c_{t-1} + I_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.6}$$

$$h_t = O_t \odot \tanh(c_t). \tag{3.7}$$

$$o_t = f(W_o h_t + b_o) \tag{3.8}$$

Where σ is a sigmoid function, x_t is an input vector at time t, h_t is a hidden state vector at time t, W is an input to hidden weight matrix, U is a hidden to hidden weight matrix, and b_t is a bias term.

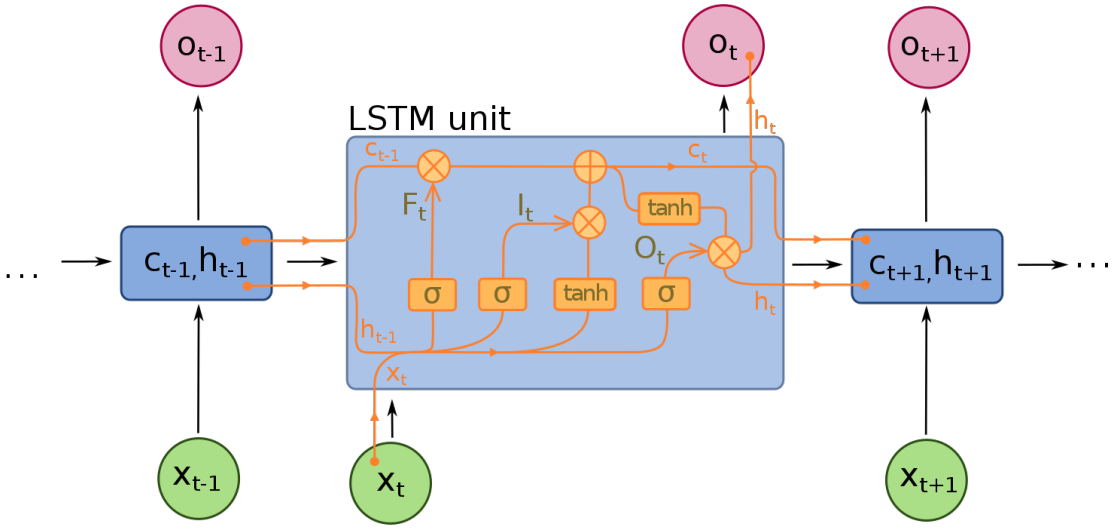

Figure 3.2: LSTM Cell Architecture.

3.4 Gated Recurrent Unit (GRU)

GRU is a variant of LSTM which was introduced by K.Cho [22], [13]. GRU is basically an LSTM without an output gate, which therefore fully writes the contents from its memory cell to the larger net at each time-step. Its internal structure is simpler and therefore considered faster to train as there are fewer computations needed to make updates to its hidden state. GRU

has two gates: reset gate r, and update gate z. Intuitively, the reset gate determines how to combine the new input with the previous memory cell, and the update gate defines how much of the previous memory cell to keep. The gates for a GRU cell are illustrated in Figure 3.3.

Equations governing the operations of GRU architecture are given below:

$$Z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.9}$$

$$labeleqgru2R_t = \sigma(W_R x_t + U_R h_{t-1} + b_R) \tag{3.10}$$

$$labeleqgru3h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot \tanh(W_h x_t + U_h(R_t \odot h_{t-1}) + b_h \tag{3.11}$$

Where R_t is the reset gate, Z_t is the update gate, h_t is the activation function and h˜ t is the candidate activation. $\odot$ is an element-wise multiplication, and σ is the logistic sigmoid function. W and U are denoted as learned weight matrices.

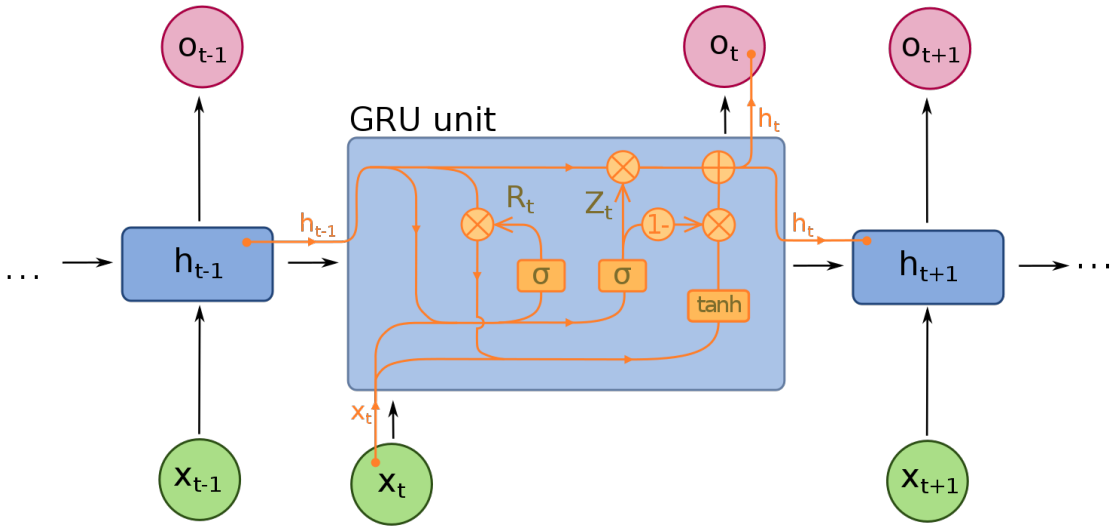

Figure 3.3: GRU Architecture.

3.5 Bi-Directional LSTM (BLSTM)

Bi-Directional RNN is also introduced to overcome the limitation of RNN [36]. This architecture can be trained using all available input information in the past and future of a specific time frame as illustrated in Figure 3.4. In other words, stacking two RNNs together in which the input sequence is fed in normal time order for one network as equation 3.12, and in reverse time order for another as equation 3.13. The outputs of the two networks are usually concatenated at each time-step.

Bi-Directional LSTM equations as below:

$$\vec{h}_t = f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b}) \tag{3.12}$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t-1} + \overleftarrow{b}) \tag{3.13}$$

$$y_t = g(U[\overrightarrow{h}_t; \overleftarrow{h}_t] + c) \tag{3.14}$$

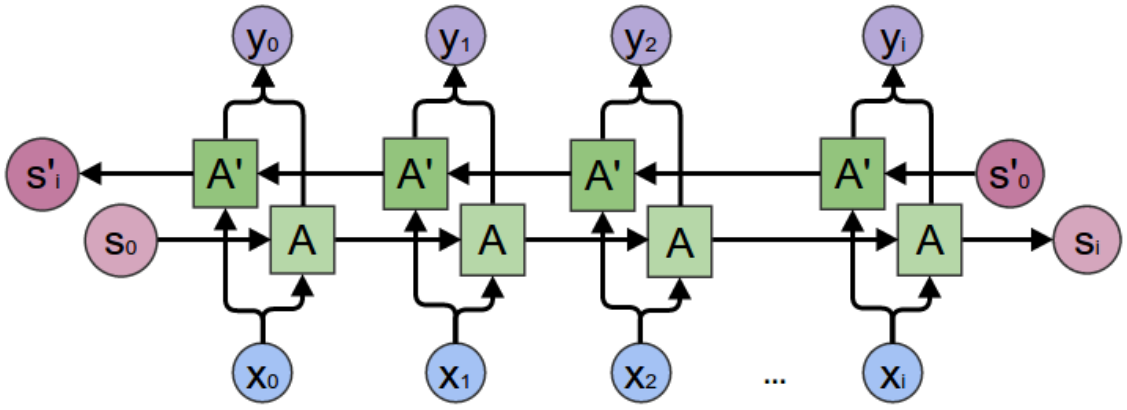

Figure 3.4: Bi-Directional LSTM Architecture.

3.6 Dynamic-RNN LSTM/GRU

Dynamic Recurrent Neural Network (DRNN) has feedback connections that dynamically perform the unrolling of the LSTM cell over each time-step. DRNNs allow for variable sequence lengths. Specifically, the state is an internal detail that is passed from time-step to another. DRNN can handle substantially longer sequences, as it handles different sequence lengths per batch, and faster graph building times, since it uses an internal loop. For a single layer LSTM model of 2048 units and batch size 256, DRNN LSTM can handle a sequence of length 256 while the normal LSTM runs out of memory at 128 [37].

3.7 Random Forest (RF)

Random Forest (RF) is a tree-based algorithm used to obtain estimates of feature importance [38]. The RF algorithm builds a large number of simple classifiers using randomly chosen features and therefore, a subset can be created of the most important features [39]. Properties of the RF as Jaiswal et al. [40] listed in their literature;

- It has been considered as unexcelled algorithm in accuracy.
- It is very efficient on huge data sets even with hundreds and thousands of input variables without overfitting and there is no requirement for data pruning.
- It is applied for the feature subset selection and missing data imputation and performs very efficiently.
- An internal unbiased estimate of the generalization error is produced by the RF algorithm in the process of forest construction.
- The generated forest can perform well for the future addition data.

The RF classifier ability was employed to rank the importance of the features set to the target variables. Only those variables based on the maximum importance levels were selected. Those with low values of the importance which were irrelevant to the learning model were discarded since it would negatively impact accuracy.

3.8 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimension-reduction algorithm that can be used to reduce a large number of features to a smaller set that still contains most of the information of the large set [41]. PCA selects a subset of features, based on which original features have the highest correlations with the principal component. It is considered an unsupervised learning algorithm that is affected by the scale of the data, but always aiming to find a meaningful way to flatten the data by focusing on items that are different between variables [42].

3.9 Parameters

Selecting the suitable parameters for RNN architectures can often make the difference in terms of performance as it has a significant impact on the accuracy [8]. However, little is published regarding which parameters and design choices should be evaluated or selected, making the correct parameters for obtaining the best performance left to experience or trial and error. The parameters used in designing the RNN architecture are learning rate, hidden layers, number of neurons in the hidden layers (hidden units), and number of time-steps.

3.9.1 Learning Rate

The learning rate is a parameter that controls how much adjustment must be made to the weights of a given network with respect to the loss gradient. For example, a far too large learning rate or dropout rate will prevent the model from learning effectively. The learning rate must be in the range of 0.01-0.1 to yield satisfactory results [43].

3.9.2 Hidden Layers

The initial design of any RNN model is comprised of a single hidden layer followed by a standard feedforward output layer [8]. Usually, the number of hidden layers to be used is based on the size of the dataset and the dimensions.

3.9.3 Hidden Units

Hidden units are the number of neurons in the RNNs hidden layer. If a higher number is in possession , the network becomes more powerful, however, the number of parameters to learn also rises. This requires more time to train the prediction model.

3.9.4 Time-Steps

Time-steps are associated with how many steps back in time backpropagation uses when calculating gradients for weight updates during training. The number of time-steps affects learning. For example, high time-steps (over 100) typically means convergence is slower, while low time-steps (a range of 8-32) means convergence is faster. For intrusion detection, time-steps play a crucial role since the number of time-steps that are required to backpropagation would impact identifying the correct patterns.

3.10 Evaluation Matrices

For evaluation purposes, Precision (P), Recall (R), F-measure (F) and Accuracy (ACC) metrics are used. These metrics are calculated by using four different measures, true positive (TP), true negative (TN), false positive (FP) and false negative (FN):

- TP: the number of anomaly records correctly classified.
- TN: the number of normal records correctly classified.
- FP: the number of normal records incorrectly classified.
- FN: the number of anomaly records incorrectly classified.

Accuracy (AC): the percentage of true detection over total traffic records,

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.15}$$

Precision (P): the percentage of predicted anomalous instances predicted are actual anomalous instances,

$$P = \frac{TP}{TP + FP} \tag{3.16}$$

Recall (R): the percentage of predicted anomalous instances versus all the anomalous instances presented,

$$R = \frac{TP}{TP + FN} \tag{3.17}$$

Chapter 4

Experimental Results

In this research the most current framework is used, Tensorflow [44], in implementing a model for each architecture. The experiments were performed on a Desktop machine with an Intel Core i7-4820K CPU @ 3.70GHz and 23.5 GB of memory. Four models were developed for each architecture: vanilla LSTM, GRU, BLSTM, Dynamic LSTM and GRU, and Skip-RNN. The experiments were designed to evaluate the performance of each model on the full KDD Cup'99 dataset in terms of accuracy and training time required for each model.

The experiment was executed using the developed prediction model (LSTM, GRU, BLSTM, and Dynamic LSTM/GRU) on the KDD Cup'99 Dataset. First, by preprocessing the dataset by scaling the features and converting non-numerical features to numerical values. Second, by implementing feature selection using two different algorithms for feature selection: RF and PCA, for the purpose of evaluating the best technique with the KDD Cup'99 dataset. Two models, LSTM and GRU, were evaluated to determine which algorithm to move forward with evaluating the rest of the experiment models. Third, by splitting the dataset into two sets: 80% for training and 20% for testing. The prediction model was run for both training and testing classifiers about 10 times, recording the best values of all readings. Finally, the accuracy of all

prediction models and the time required to train the models was logged. All the matrices were calculated including True False Alarm Rate (Recall), False Alarm Rate (FAR), Efficiency and Precision. All matrices are shown in equations (3.15, 3.16 3.17). The experiment process is illustrated in Figure 1.1.

4.1 Dataset Description

The study sample was conducted from the Third International Knowledge Discovery and Data Mining Tools Competition (KDD Cup) 1999 [9]. The dataset was a collection of simulated raw TCP dump data over a period of nine weeks on a local area Network. The three versions of the KDD Cup'99 IDS datasets are the full KDD dataset, corrected KDD, and 10% KDD as shown in Table 4.1. Among these three, the 10% KDD dataset is the most used in literature. However, the experiments in this research were conducted on the full dataset (18M; 743M uncompressed) to obtain a more realistic view. The training data was collected for seven weeks and testing data was collected for two weeks. The entire dataset contains 39 attacks which are categorized into four classes: Probe, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L). There are 41 features in addition to one class label for every record "normal" or "attack type". A complete listing of the 41 features defined for KDD Cup'99 dataset is given in Appendix A.

Table 4.1: KDD Cup 1999 Datasets (Number of Samples)

KDD Dataset	Total	DoS	Probe	R2L	U2R	Normal
Whole KDD	4,898,430	3,883,370	41,102	1,126	52	972,780
Corrected KDD	311,029	229,853	4,166	16,347	70	60,593
10% KDD	494,020	391,458	4,107	1,126	52	97,277

4.2 Data Preprocessing

The purpose of data preprocessing is mainly to transform the raw input data to the proper format for the training model. The steps involved in data preprocessing are:

- Dropping duplicate records
- Dropping labels to a different dataset to be used for training RNN classifier.
- Converting Categorical data to Numerical with one-hot vector to fit the training model. Then select float64 as datatype.
- Scaling and normalizing the dataset, scaling the features so the lowest rank is 0 and the highest rank is 1.
- Coding “0 1” as an attack and “1 0” normal.
- Splitting the dataset into training dataset and testing dataset with a ratio of 8:2.

4.3 Feature Selection

The feature selection mechanism helps to identify and remove non-essential, irrelevant and redundant variables from data that has less of an effect on the accuracy. In this context, feature selection usually address what is the best representation of the data to learn a solution to the underlying problem. If this isn’t done, it could negatively impact the accuracy of the prediction model.

4.3.1 RandomForest (RF)

The "RandomForest (RF)" classifier was applied for feature selection based on a previous study that demonstrates the efficiency of the RandomForest algorithm with KDD Cup'99. RF is a tree-based algorithm used to obtain estimates of feature importance [38]. The algorithm was run on the dataset, which ranked 41 features based on importance presented in Figure 4.1. Thereafter, the top 12 features were selected for the experiment as illustrated in Table 4.2

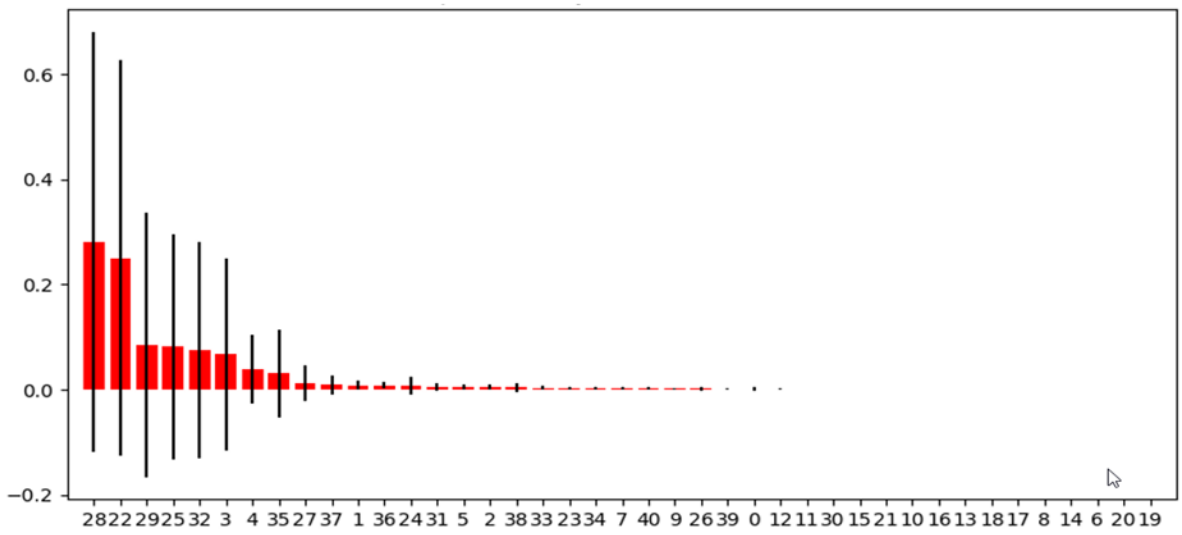

Figure 4.1: Feature Selection Based on RF. (y-axis) shows Feature Importances and (x-axis) shows Feature IDs.

4.3.2 Principal Component Analysis (PCA)

First, the mean value was calculated for each variable and subtracted the mean for each value of the same variable; then calculated the correlation matrix, then the eigenvalues and eigenvectors of the matrix, arranged the eigenvalues in a descending order and chose the top 12 features as illustrated in Table 4.3 and Figure 4.2. Their corresponding eigenvectors were used

Table 4.2: Top 12 Selected Features Based on the RF classifier

No	Feature Name	Feature ID
1	srv_rerror_rate	28
2	is_guest_login	22
3	same_srv_rate	29
4	serror_rate	25
5	dst_host_count	32
6	service	3
7	flag	4
8	dst_host_diff_srv_rate	35
9	rerror_rate	27
10	dst_host_srv_diff_host_rate	37
11	duration	1
12	dst_host_same_src_port_rate	36

as a characteristic vector matrix. Finally, the corresponding data was projected to the selected eigenvectors and ended with the "processed" dataset. If a dataset with sample M is available, and variable N, the original (mean-normalized data) data is M*N, and the correlation matrix is N*N. The top K eigenvalues were chosen as the selected feature, therefore these eigenvectors consist of characteristic vector matrix set N*K. The relationship between them is illustrated in equation 4.1

$$FinalData(M * K) = O(M * N) * E(N * K) \tag{4.1}$$

Where O is the original data, E is the eigenvector, M is the number of samples, N is the number of variables, and K is the top selected eigenvalues.

As presented in this section, the features were selected based on the RF algorithm which demonstrated a better presentation to the dataset and a higher accuracy in predicting whether the samples were anomaly or not. The top 12 features were selected since adding any extra features would not impact the accuracy. It is worth mentioning that the two algorithms had five common features among the 12 features listed by ID as follows: 22, 25, 27, 29 and 37. The result of each model using RF is presented in detail in the Results, Analysis and Discussion section.

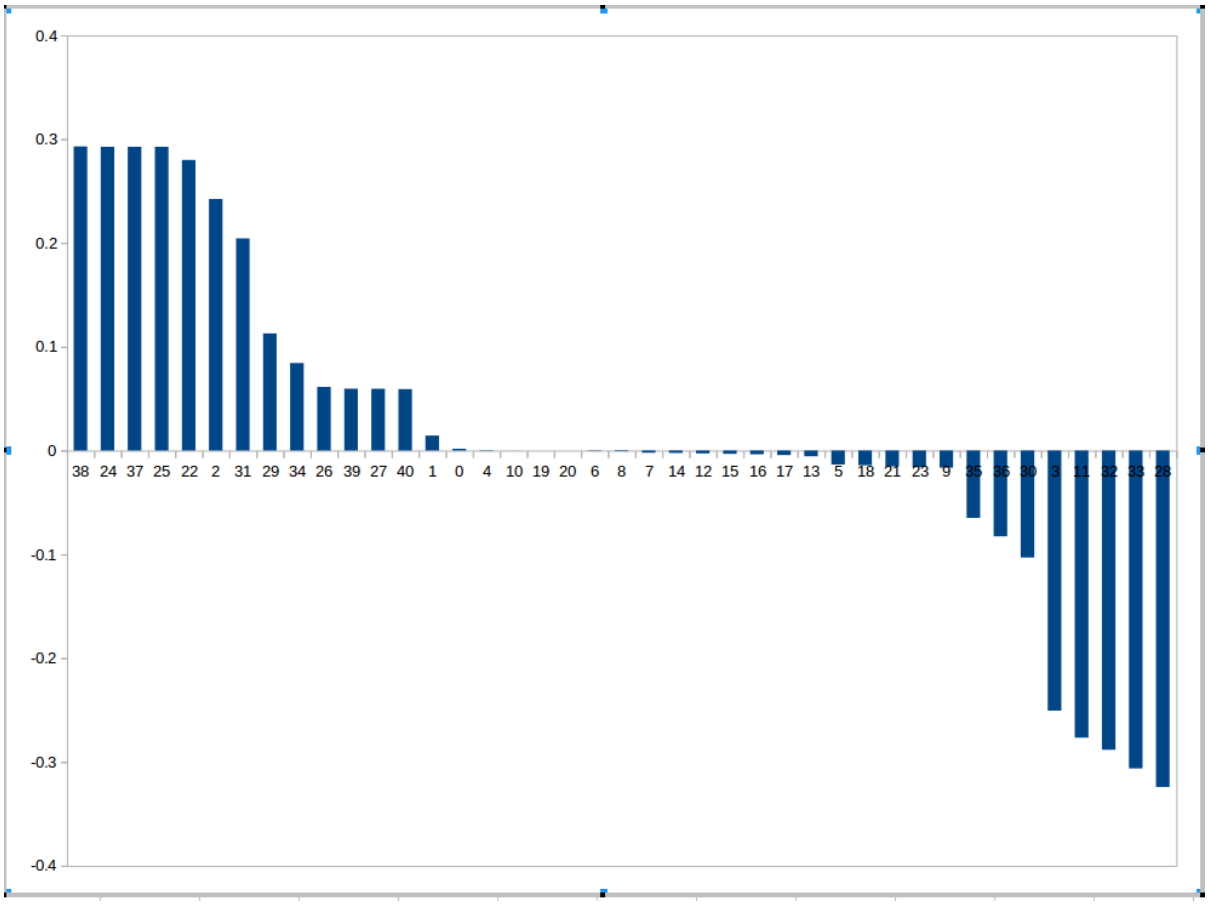

Figure 4.2: Features Selection Based on the PCA Classifier.

Table 4.3: Top 12 Selected Features Based on the PCA classifier

No	Feature Name	Feature ID
1	dst_host_serror_rate	38
2	srv_count	24
3	dst_host_srv_diff_host_rate	37
4	serror_rate	25
5	is_guest_login	22
6	protocol_type	2
7	srv_diff_host_rate	31
8	same_srv_rate	29
9	dst_host_same_srv_rate	34
10	srv_serror_rate	26
11	dst_host_srv_serror_rate	39
12	rerror_rate	27

Chapter 5

Results, Analysis, and Discussion

5.1 Introduction

The experiment spanned two phases. The first phase illustrates the difference between two features selection algorithms, RF and PCA for each LSTM and GRU. The impact of each algorithm was identified based on the prediction accuracy. The second phase carried the experiment using RF algorithm as the feature selection algorithm for rest of the RNN architectures since the outcome of the first phase of the experiment shows that RF captures more significant features over PCA.

5.2 Phase I: Feature Selection

The first phase begins with training the LSTM and GRU, and setting up specific parameters with the RF features selection algorithm. The set of parameters are set as: training rate 0.01, hidden layers 10, and backpropagation (working our way backwards through the network) 5, with different cycle values for training. The same experiment setup was repeated using the

PCA algorithm for features selection. The experiment was executed around 10 times for each algorithm with different sets of parameters combinations. LSTM and GRU were trained at different training cycles: 50, 100 and 300 as illustrated in Table 5.1. The goal was to obtain the best case of prediction accuracy and training time. Based on the experiment the observations are as follows:

- The best accuracy of LSTM and GRU at each training cycle with RF was achieved. It shows that at 50 cycles, the LSTM and GRU scored an accuracy of 97.86% and 96.59%, respectively. Whereas both models scored 64% and 67.97% with PCA.
- At training cycles with the value 300, LSTM and GRU cored an accuracy of 97.54% and 97.57% with PCA. However, this still did not outperform the accuracy recorded by RF which were 99.00% and 98.89%.
- Based on the result, the rest of the experiment carried on using RF for feature selection in training LSTM, GRU, BLSTM, and DRNN LSTM/GRU.

Table 5.1: LSTM and GRU Accuracy
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 10

Training Cycles	50	100	300
RandomForest			
LSTM	97.86%	98.36%	99.00%
GRU	96.59%	97.21%	98.89%
PCA			
LSTM	64.34%	96.41%	97.54%
GRU	67.97%	96.41%	97.57%

5.3 Phase II: RNN Architectures for IDS

In phase two, RNN architectures: LSTM, GRU, BLSTM, and DRNN LSTM/GRU are used to be trained to detect anomalies. The implemented models were developed using Python and

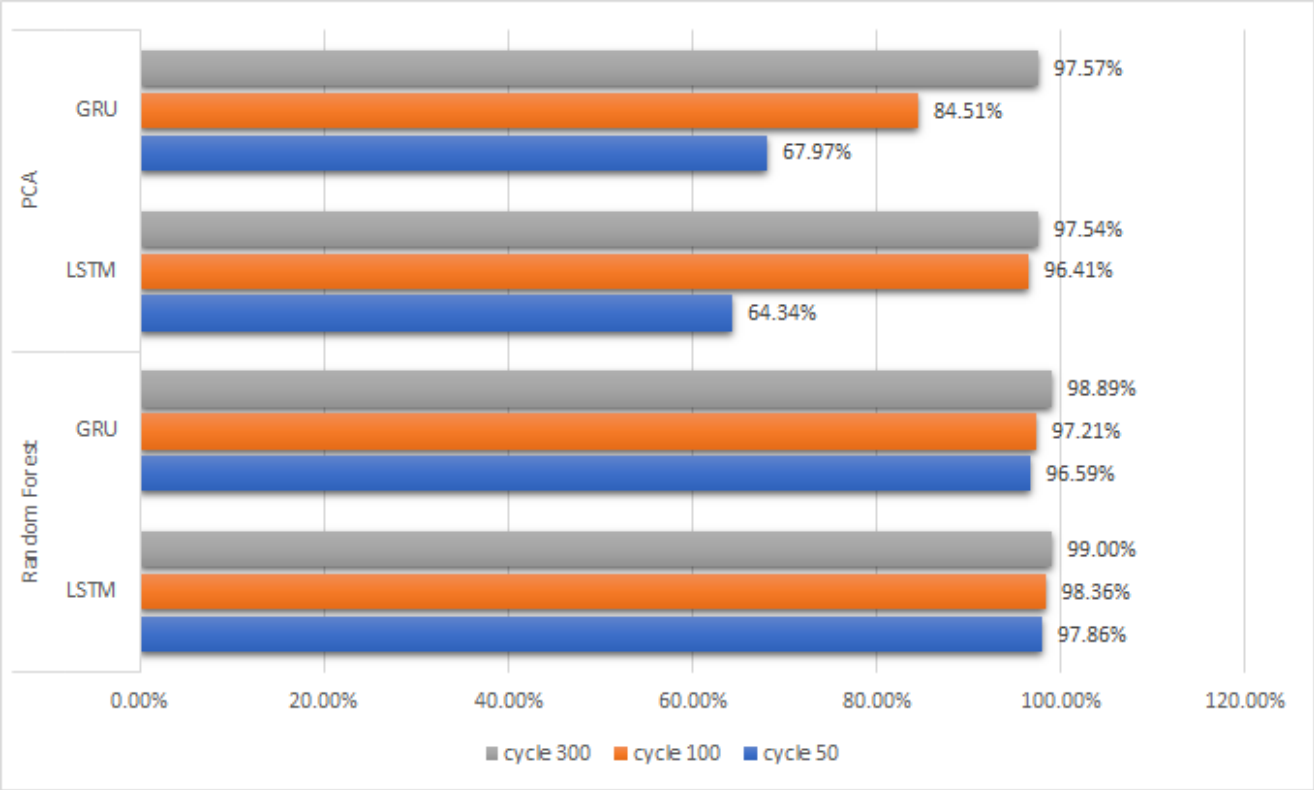

Figure 5.1: LSTM and GRU Accuracy Comparison between RF & PCA at Different Training Cycles.

TensorFlow platforms [44] to show the capability of each model to learn the definition of being normal and anomalous from labeled datasets. Each model on the KDD Cup'99 dataset was evaluated as previously mentioned. The first model was a vanilla LSTM, which was trained and evaluated, as all RNN models selected for this experiment. For each model the best defaulted value parameter setup was identified to begin, tuning each model to achieve the best performance and the highest accuracy. Several runs were conducted with different values as shown in Table 5.2, such as learning rate, training cycle, time-step and hidden layers. The same process was followed for all models, including adding certain parameters such as batch size for BLSTM since the architecture of the model required passing the data into batches. Batch size limits the number of samples to be shown to the network before a weight update can be performed. Due to the limitation of available memory, the desired parameter values could not be

achieved for hidden layers and time-steps.

Table 5.2: Parameter Values

Parameter Name	Value	Note
Learning Rate	0.01	-
Training Cycle	100 / 500 / 1000	-
Hidden Layers	25/50	-
Time-step	5/10	-
Batch Size	512	Used Only for Skip-RNN and BLSTM

5.3.1 Long Short-term Memory (LSTM)

First, a vanilla LSTM model was developed, and then determined through the experiment what the suitable learning rate was for this model. The learning rate is one of the most important parameters to be tuned due to its impact on the training model for faster and effective training. It is important to not overfit the training model. The experiment was run with three different learning rates: 0.0001, 0.001, and 0.01. After running the experiment several times at value 100, the training cycles results show that the learning rate 0.001 gives the best loss value, which then decreases during training to allow more weight updates. A learning rate of 0.0001 did not allow the model to converge as illustrated in Figure 5.2.

LSTM model was trained at three different cycles; 100, 500, and 1000. The accuracy, precision, recall, FAR and time required for the model to be trained for each training cycle was calculated. As shown in Table 5.3, LSTM shows a higher accuracy of 99.43% at 500 training cycles. It was noted that adding more training cycles did not result in an increase in accuracy prediction, as shown in in Figure 5.3. With regards to the training time, this took 10648.93 seconds for the LSTM 500 cycle, as per Figure 5.4.

Table 5.3: Vanilla LSTM
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50.

Training Cycles	Accuracy	Precision	Recall	FAR	Time (sec)
100	98.85%	98.71%	98.67%	1.00%	2354.93
500	99.43%	99.21%	99.48%	0.61%	10648.93
1000	99.25%	99.18%	99.10%	0.64%	20942.11

5.3.2 Gated Recurrent Unit (GRU)

The same steps were followed as the LSTM model to train the GRU model, with a learning rate once more at 0.01, as shown in Figure 5.5. The GRU model was trained at three different training cycles; 100, 500, and 1000, calculating the accuracy, precision, recall, FAR and the time required for the model to be trained for each of the training cycles. As shown in Table 5.4, the GRU model shows a higher accuracy of 99.34% at the 1000 cycle, as shown in Figure 5.6. GRU required more learning cycles to outperform LSTM. However, it is important to keep in mind that the training time was less than LSTM due to the fact that GRU architecture consists of two gates only. GRU has fewer parameters, and took 16654.21 seconds as shown in Figure 5.4.

Table 5.4: GRU
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50.

Training Cycles	Accuracy	Precision	Recall	FAR	Time (sec)
100	98.68%	98.77%	98.18%	0.94%	1903.09
500	99.06%	99.05%	99.78%	0.73%	8579.35
1000	99.34%	99.19%	99.31%	0.63%	16654.21

5.3.3 Bi-Directional LSTM (BLSTM)

BLSTM results show that the model didn't perform as well as the rest of the models. The accuracy at each training cycle below 90% is illustrated in Table 5.5. The one significance of the model is with respect to the training time. It shows a significant improvement in the training

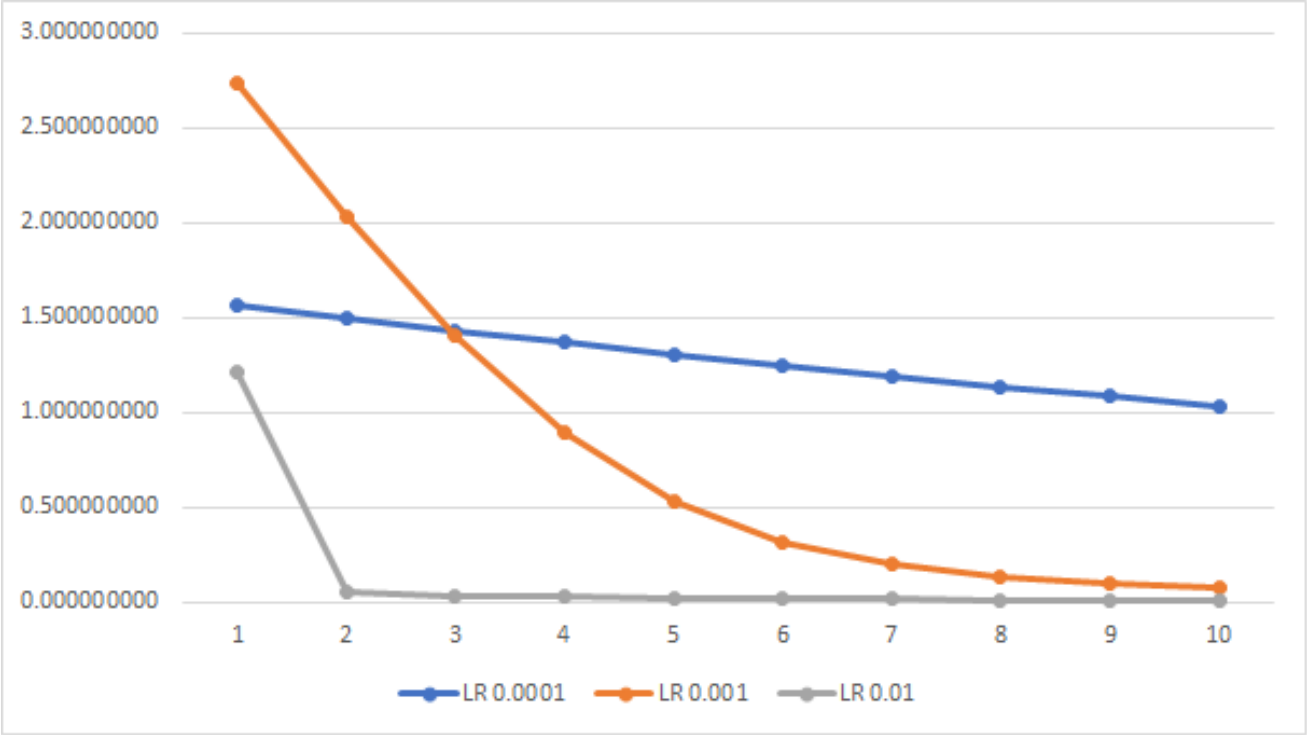

Figure 5.2: Learning Rate Cost for LSTM
with Training Cycle: 100, Time-Step: 5, and Hidden Layers: 50.

time, for instance, at the 500 training cycle the model required only 190 seconds.

Table 5.5: BLSTM
Learning Rate 0.01, Backpropagation: 5, and Hidden Layers: 50.

Training Cycles	Accuracy	Precision	Recall	FAR	Time (sec)
100	84.99%	18.25%	72.03%	25.80%	190
500	82.20%	5.3%	2.1%	3.0%	250.3
1000	43.60%	14.40%	14.40%	78.0%	143.52

5.3.4 Dynamic-RNN LSTM/GRU

DRNN architecture had two implementations: LSTM and GRU. DRNN architecture uses variable length sequences, which compute the shape of the input sequence length, whereas other architectures have a fixed size of input sequence length. It relies on the padding technique

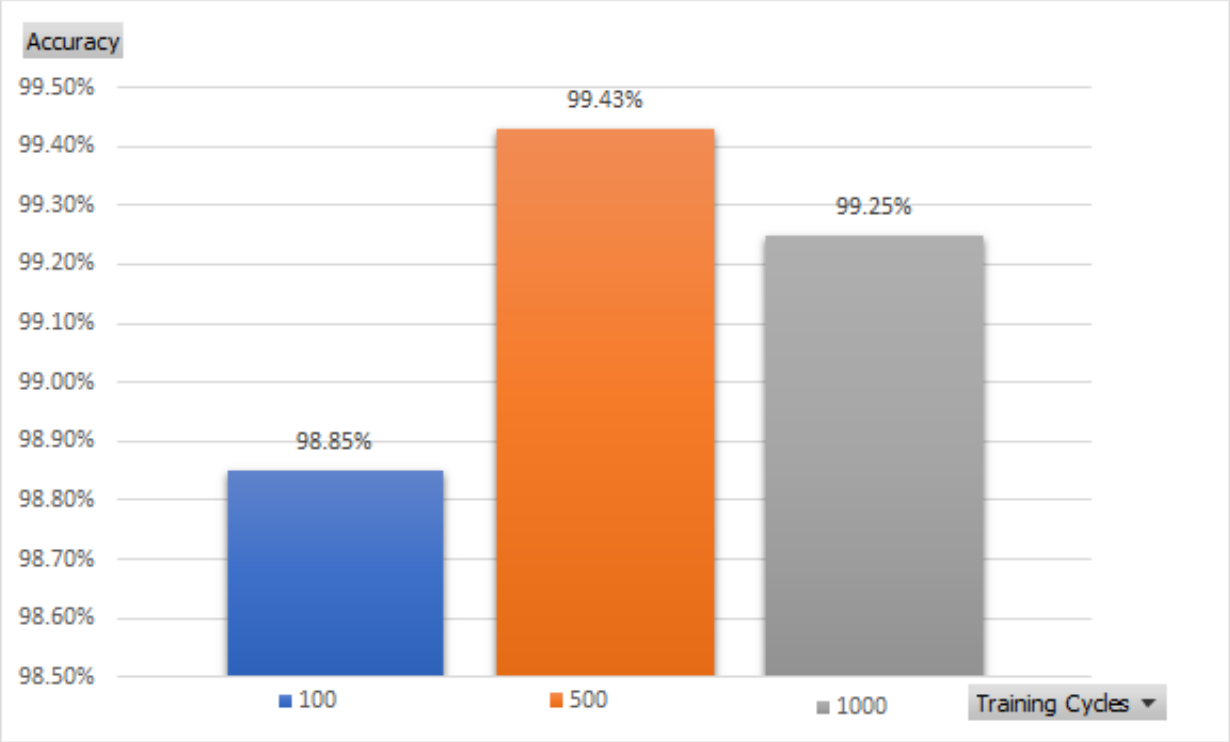

Figure 5.3: LSTM Accuracy
for each Training Cycle 100, 500, and 1000.

by having a vector holding the sequence lengths, that can be passed to the DRNN model. The experiment followed the same steps by obtaining the best learning rate for DRNN which is 0.01, as shown in Figure 5.8 for LSTM and Figure 5.9 for GRU. Moreover, the accuracy value of each model was presented at each learning rate as seen in Table 5.6. Each model, LSTM and GRU, was trained at the same training cycles: 100, 500, and 1000. The model parameters were set up with 50 hidden layers, and 5 time-steps. The results showed the best accuracy as 99.27% for DRNN LSTM at the 100 training cycle, where DRNN for GRU outperformed DRNN LSTM, scoring 99.34%. However, it dropped significantly at 1000 training cycles scoring 90.24%, whereas DRNN LSTM maintained its high accuracy at 99.23% for the same training cycle. DRNN LSTM showed its best performance at 500 training cycles, whereas DRNN GRU scored the highest accuracy at 100 training cycles as shown in Figure 5.10. With

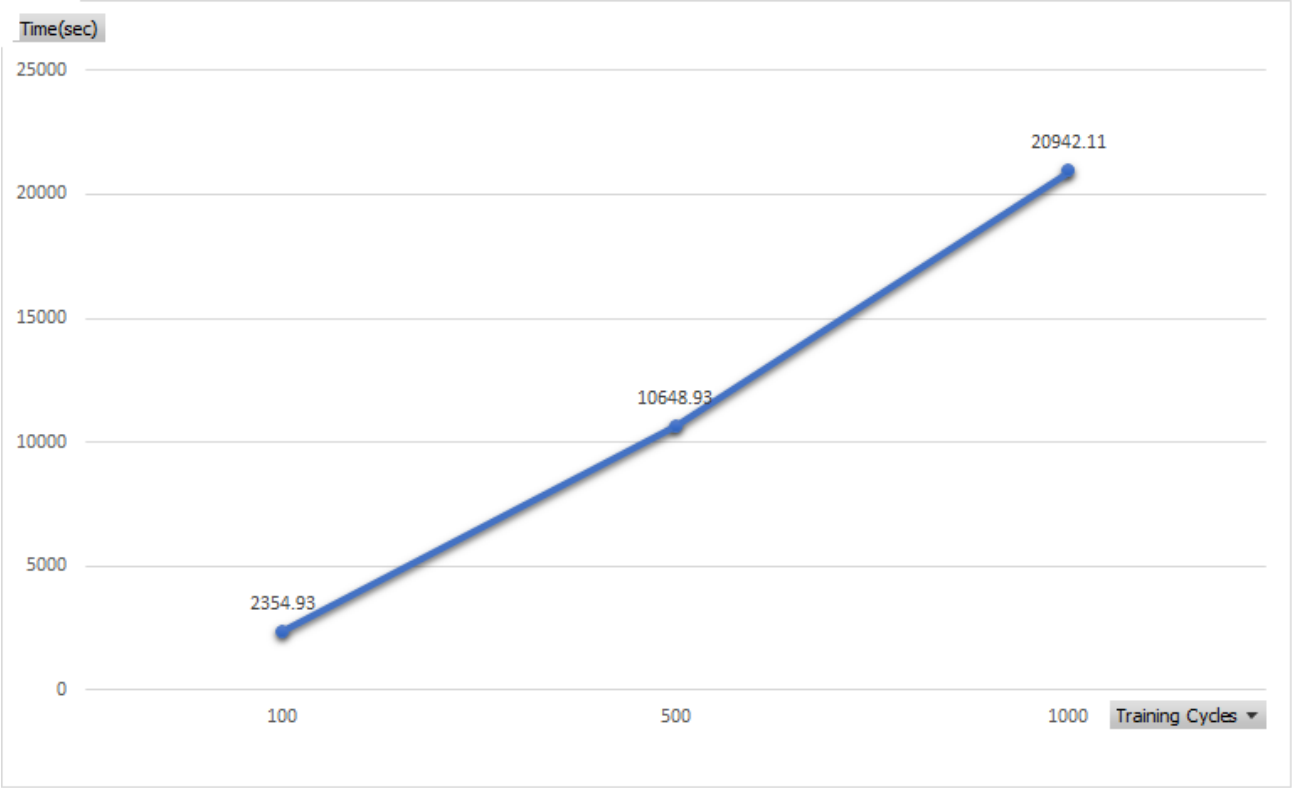

Figure 5.4: LSTM Training Time
for each Training Cycle 100, 500, and 1000.

regards to the time taken to train the model, DRNN LSTM required 6154.80 seconds and DRNN GRU required 2072.89 seconds. Both of the models' accuracy, time and other matrices are illustrated in Table 5.7 and Table 5.8.

Table 5.6: DRNN Accuracy for Each Learning Rate

Model	**LR 0.0001**	**LR 0.001**	**LR 0.01**
DRNN LSTM	84.65%	98.20%	99.08%
DRNN GRU	86.49%	97.49%	98.86%

5.3.5 Overall Analysis

In this experiment, the dataset with intrusion attacks containing 4,898,431 samples with 262,178 attacks was used. A similar series of experiments were run on all selected architectures

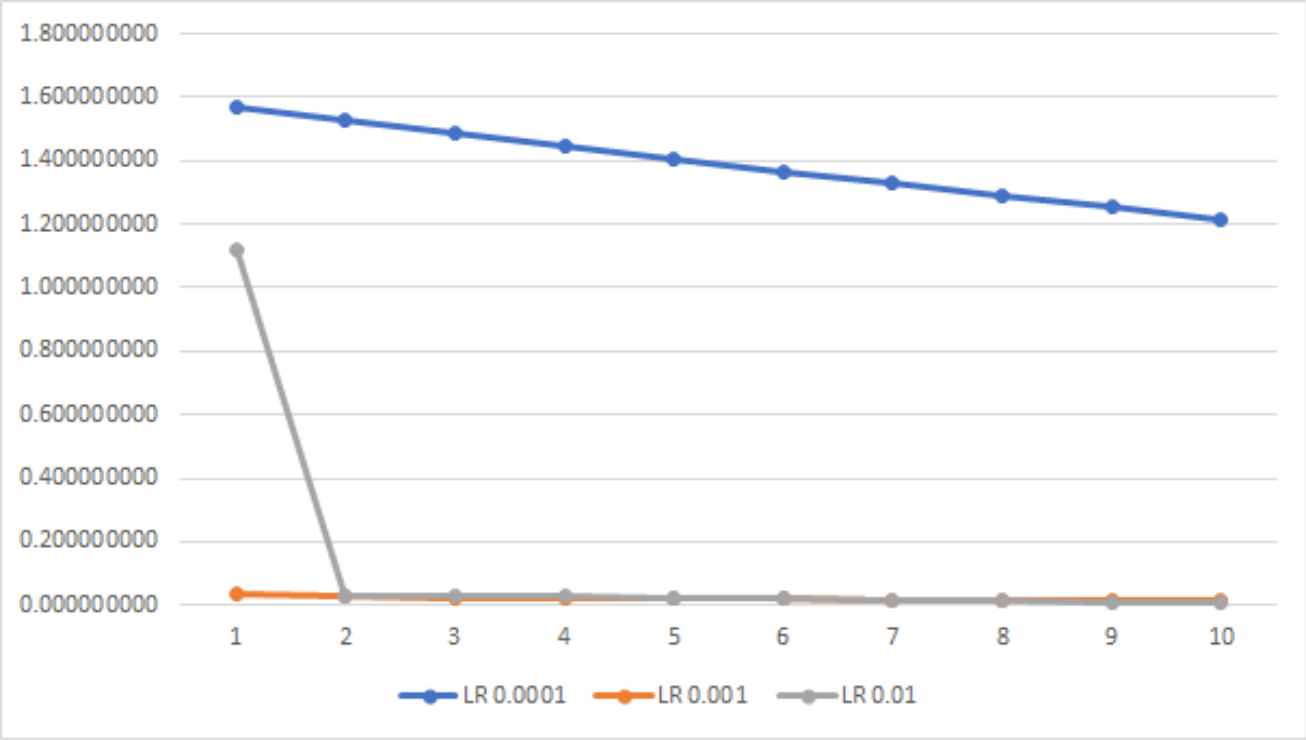

Figure 5.5: Learning Rate Cost for GRU
with Training Cycle: 100, Time-step: 5, and Hidden Layers: 50.

with a slight change in the parameters due to some architecture requirements. Each algorithm was trained at different training cycles as in Table 5.9, and the observations are as follows:

- The learning rate is the single most important parameter, and for the KDD Cup'99 the best learning rate with the selected machine learning model is 0.01.
- DRNNs show the best performance in terms of accuracy with fewer training cycles. DRNN LSTM accuracy is at 99.27% where vanilla LSTM accuracy is 98.85%. DRNN GRU accuracy is 99.34% whereas GRU accuracy is 98.68%. This is due to the fact that the DRNNs allow for variable sequence lengths. This allow RNNs to run for the correct number of time-steps on those sequences that could be shorter than the maximum sequence length.
- DRNN best accuracy rate is at 100 training cycles. Adding more training cycles could

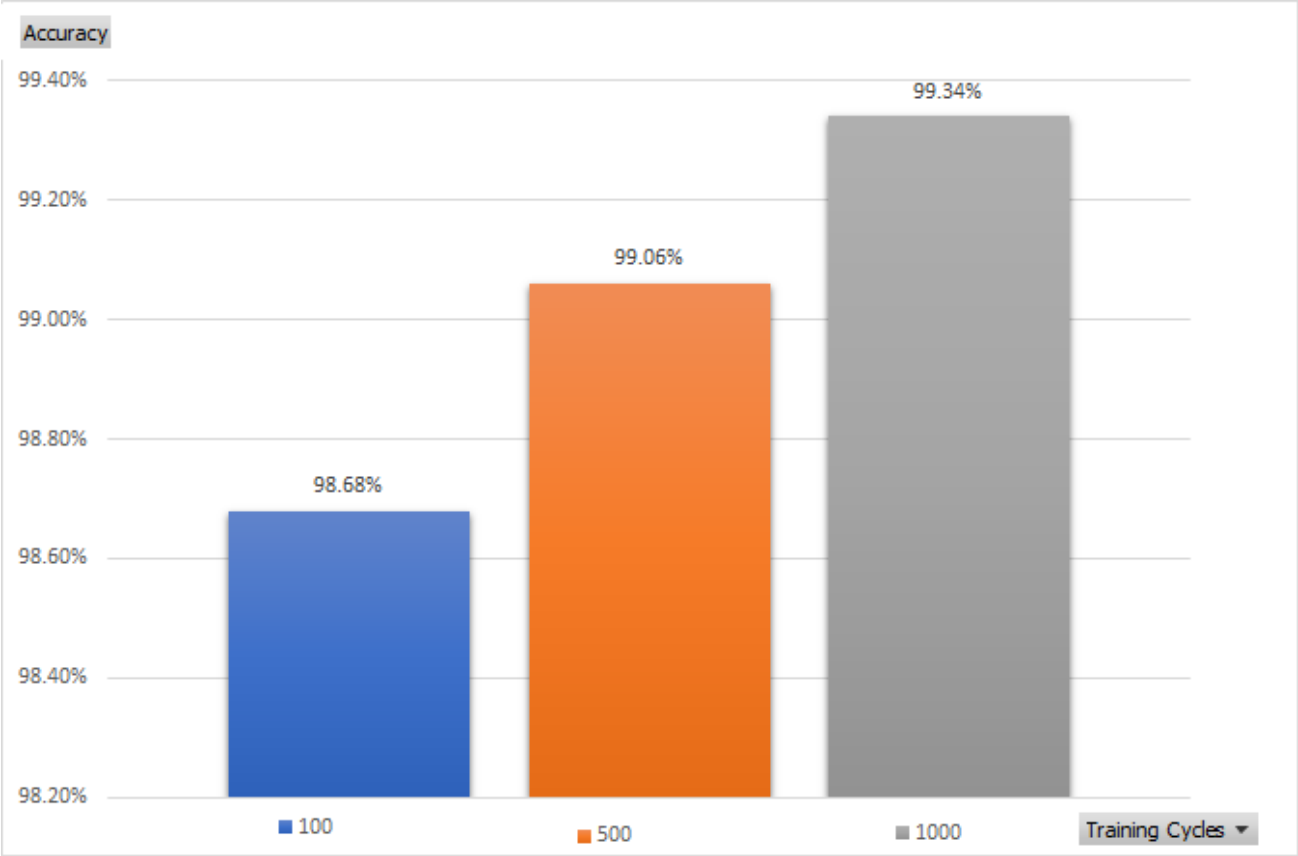

Figure 5.6: GRU Accuracy
for each Training Cycle 100, 500, and 1000.

lead to an overfitting of the model.

- Highest accuracy recorded for the standard Vanilla LSTM is 99.43% at 500 training cycles, with 10648.93 seconds. LSTM architecture is suitable for large-scale implementation.

- Meanwhile, GRU outperformed LSTM at 1000 training cycles. GRU required more training cycle to increase its accuracy because GRU uses two gates, the update gate and reset gate. It lacks the output gate which fully writes the contents from its memory cell to the larger net at each time-step.

- Highest accuracy at 100 training cycles was DRNN GRU which scored 99.34%, then

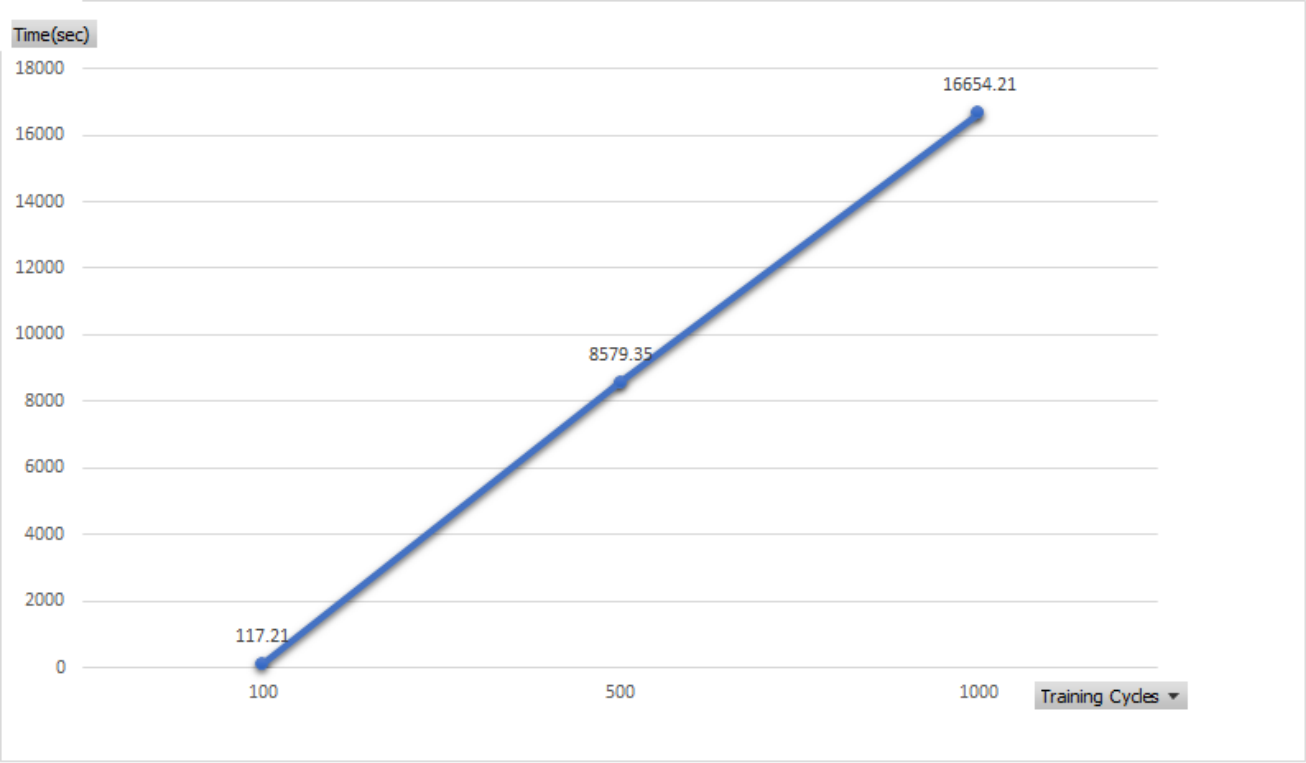

Figure 5.7: GRU Training Time
for each Training Cycle: 100, 500, and 1000.

DRNN LSTM at 99.27%. The highest accuracy for 500 Training Cycle was LSTM scored 99.43%. Finally, GRU scored the highest accuracy at 99.25%.

- GRU scored the best training time at two training cycles: 500 and 1000. All training time scores are presented in Table 5.10 and in Figure 5.11.
- Due to the limitation of available memory, the desired parameter values could not be achieved for hidden layers and time-steps (Backpropagation). This restriction did not allow to test the algorithm as expected, leading to trade off between hidden layers and time-steps. The best set value was 50 for hidden layers and 5 for time-steps.
- Due to the GRU model having fewer parameters than LSTM, the GRU model proved to train faster. It required only 31.7 minutes while LSTM required 39.25 minutes.
- BLSTM required less time for training, but its accuracy rate was lower than the rest of

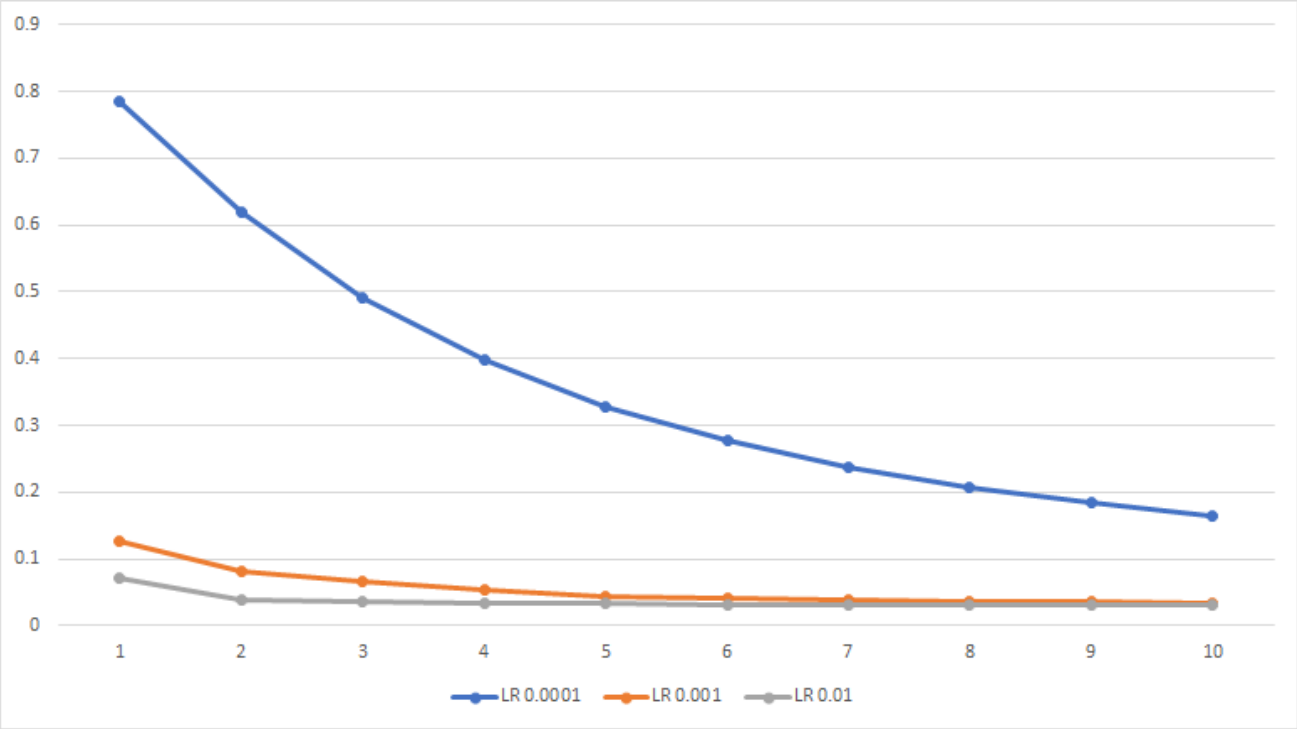

Figure 5.8: Learning Rate Cost for DRNN LSTM
with Training Cycle: 100, Time-step: 5, and Hidden Layers: 50.

the RNN's models. More investigation is required to see how to enhance the algorithm accuracy.

- Skip-LSTM could not be trained in KDD Cup'99 intrusion detection even after running several attempts with different setting parameters in the algorithm. More investigation is required to fit the data into the model in order for the algorithm to predict attacks.

- In comparison with other approaches presented in other literature using LSTM the proposed optimized LSTM in this research scored a the highest accuracy 99.43% where other approaches the highest scored was 98.95% illustrated in Table 5.11 and Figure 5.12. Where as 19,593 more attacks out of 3,925,650 have been correctly detected.

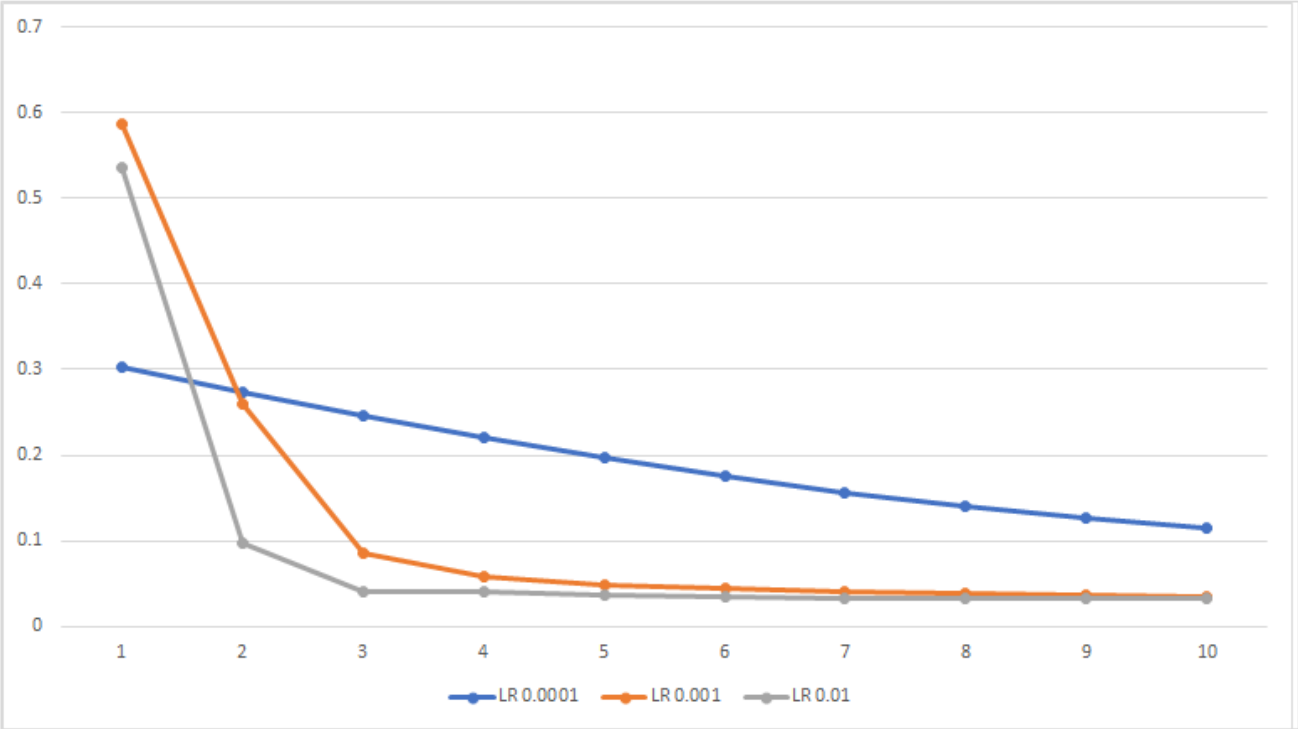

Figure 5.9: Learning Rate Cost for DRNN GRU
with Training Cycle: 100, Time-step: 5, and Hidden Layers: 50.

Table 5.7: DRNN LSTM
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50

Training Cycles	Accuracy	Precision	Recall	FAR	Time (sec)
100	99.27%	99.10%	99.23%	0.70%	2491.52
500	98.92%	99.33%	99.51%	0.50%	6154.80
1000	99.23%	99.43%	98.81%	0.436%	25284.03

Table 5.8: DRNN GRU
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50.

Training Cycles	Accuracy	Precision	Recall	FAR	Time (sec)
100	99.34%	99.20%	99.29%	0.621%	2072.89
500	99.14%	99.34%	98.22%	0.51%	5480.13
1000	90.24%	82.17%	99.13%	16.64%	20918.18

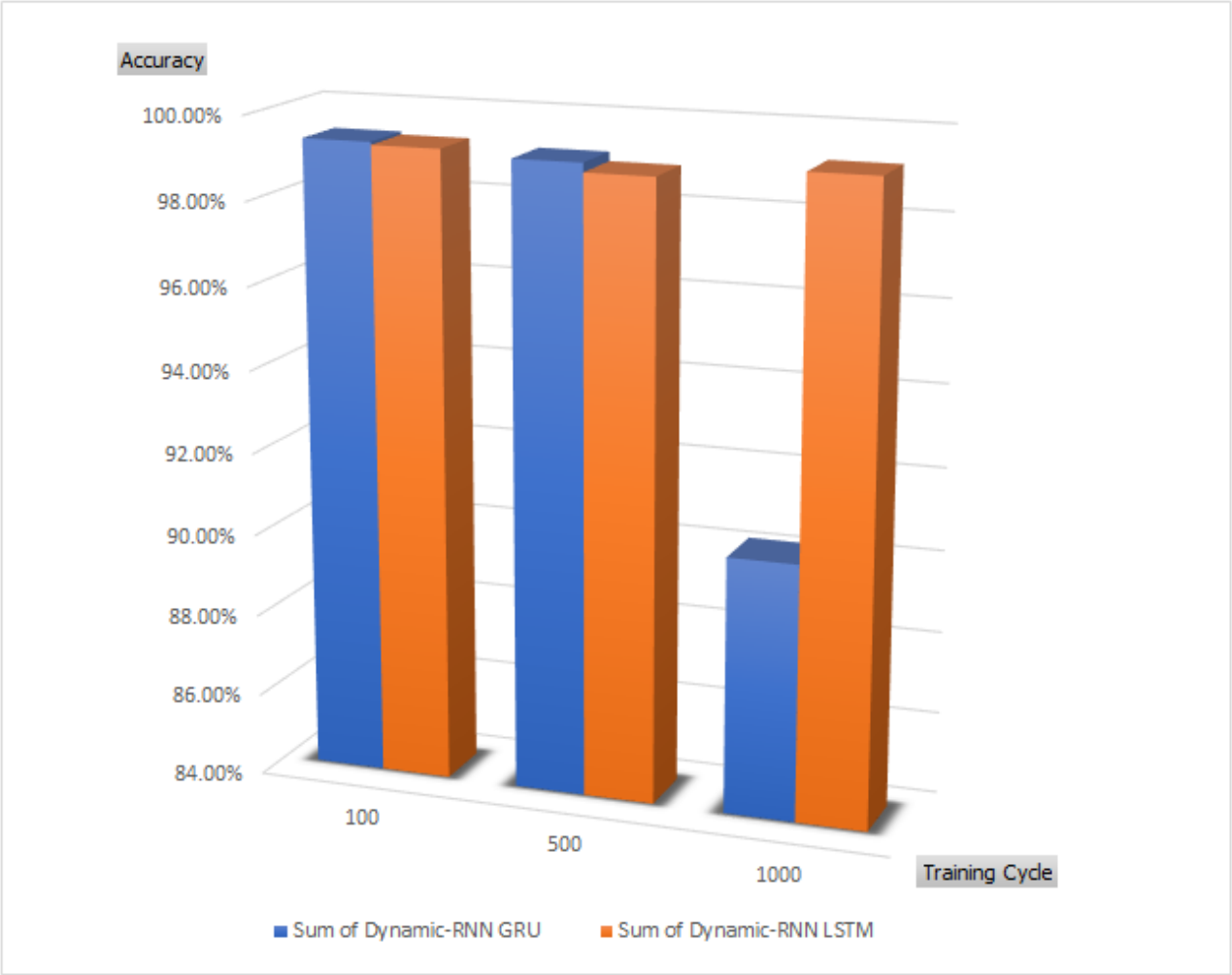

Figure 5.10: DRNN LSTM and DRNN GRU Accuracy
with Training Cycle: 100/500/1000, Time-step:5, and Hidden Layers:50.

Table 5.9: RNN Architecture Overall Comparison for Accuracy
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50

Training Cycles	**100**	**500**	**1000**
LSTM	98.85%	99.43%	99.25%
GRU	98.68%	99.06%	99.34%
BLSTM	84.99%	82.20%	43.60%
DRNN LSTM	99.27%	98.92%	99.23%
DRNN GRU	99.34%	99.14%	90.24%

Table 5.10: RNN Architecture Overall Comparison for Training Time
Learning Rate: 0.01, Backpropagation: 5, and Hidden Layers: 50

Model	**Time (sec)**	**Time (sec)**	**Time (sec)**
Training Cycles	**100**	**500**	**1000**
LSTM	2354.93	10648.93	20942.11
GRU	1903.09	8579.35	16654.21
BLSTM	190	250.3	143.52
DRNN LSTM	2491.52	6154.80	25284.03
DRNN GRU	2072.89	5480.13	20918.18

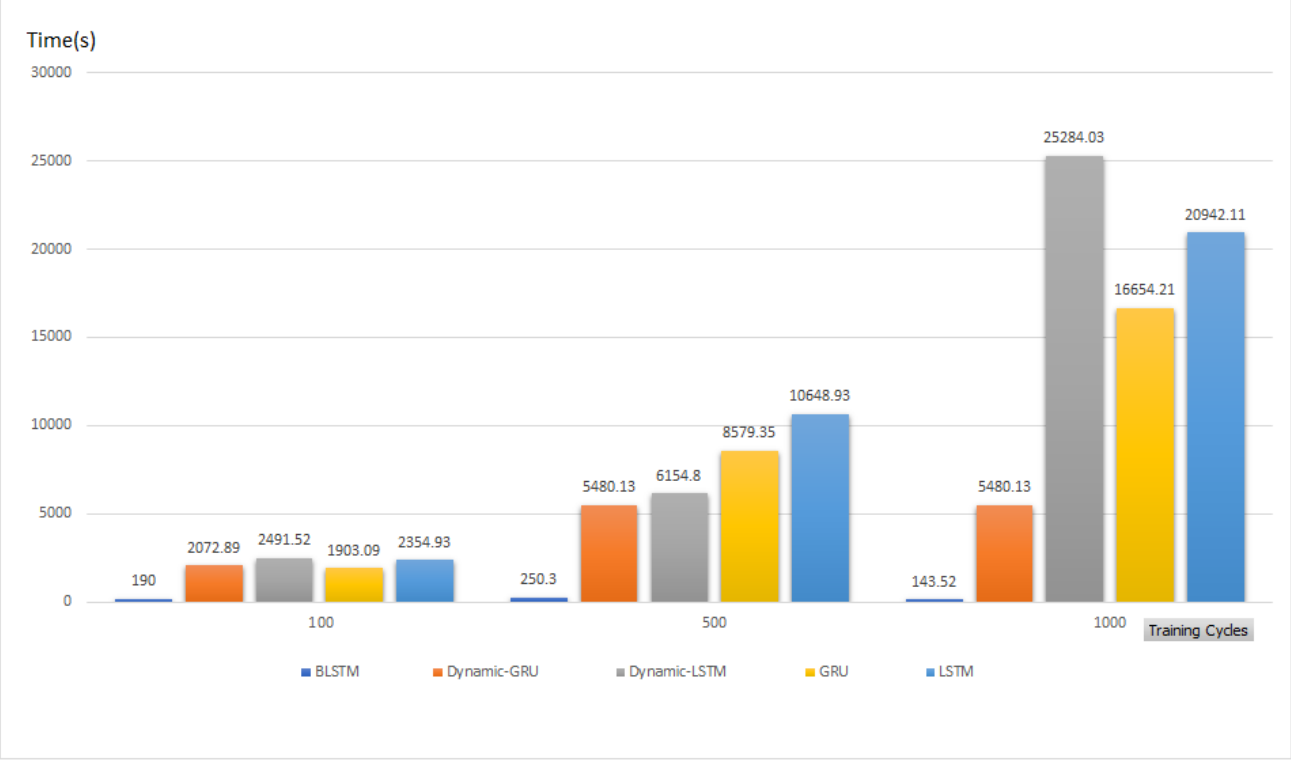

Figure 5.11: RNN Architectures over all Training Time
with Training Cycle:100/500/1000, Time-step:5 and Hidden Layers: 50.

Table 5.11: Comparison of the Optimized LSTM Model Accuracy Rate with other LSTM model proposed by other Literature Review

Approaches	Accuracy
Optimized LSTM	**99.34%**
Approach 1 [16]	98.85%
Approach 2 [17]	93.78%
Approach 3 [18]	98.85%
Approach 4 [19]	98.80%
Approach 5 [27]	98.95%

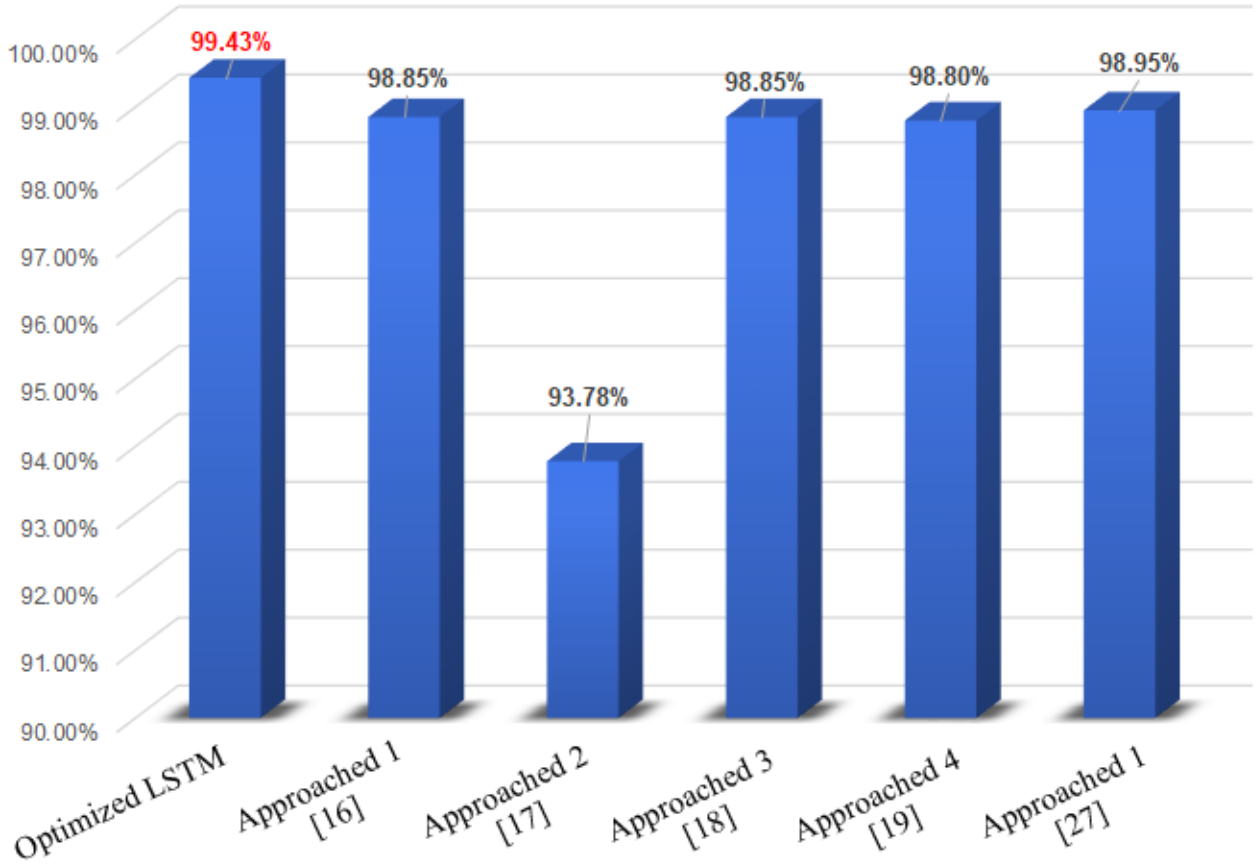

Figure 5.12: Comparison of the Optimized LSTM Model Accuracy Rate with other LSTM model proposed by other Literature Review

Chapter 6

Conclusion and Future Work

The novelty of this research stems from the fact that it is the first experiment that implements and compares RNN's architecture and offers more insight into each architecture, particularly vanilla LSTM, GRU, BLSTM, and DRNN LSTM/GRU, on an intrusion detection dataset. Most literature in the domain demonstrates the concept of using LSTM as one of the RNN architectures to improve the accuracy in predicting attacks, as well its different variants, however they only focused on one architecture for one application, comparing it with other machine learning techniques like SVM, RF, and J48. This research took the path further in understanding the architecture of each RNN algorithm, then applying it in an intrusion detection dataset. It evaluates the performance of each architecture in terms of prediction accuracy and the time required for each architecture to be trained. Moreover, this experiment is unique as it runs these architectures on the full KDD Cup'99 dataset, which contains 4,898,431 samples, rather than the commonly used KDD 10% dataset. Feature selection was performed using two different mechanisms, RF and PCA, which are suitable for intrusion detection. This has offered a clean dataset that carries all the important features. Feature selection reduced the dataset features to improve the performance of accuracy, recall, training time and false alarm rate. As part of this research, the experiment was limited in tuning the set of parameters with the goal of finding

the optimal parameters such as learning rate, hidden layers, and training cycle to improve the model's prediction accuracy and the amount of time required to be trained.

The results of this evaluation revealed that vanilla LSTM still stands up and outperforms other architectures that are supposed enhancements of the vanilla LSTM. DRNNs showed the potential to offer better performance with accuracy, however with high training cycles resulting in a tendency to take a longer time to be trained. GRU architecture showed equivalent performance with respect to the parameters such as hidden layers and time-step. GRU has fewer parameters resulting in a faster-trained model compared to LSTM. In a large-scale implementation, however, LSTM may yield better results. BLSTM offered an impressive training time, yet the accuracy did not exceed 84.99%.

For future work, the aim is to evaluate further architectures on the intrusion detection dataset. Moreover, the aim is to investigate the application of deep learning by having multiple layers and hybrid layers of different architectures in one framework, as well as deploying these techniques in IoT applications to develop robust security solutions. This is made possible with the concepts of machine learning and deep learning as IoT generates an enormous amount of heterogeneous data.

Bibliography

[1] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 2056–2063.

[2] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. Krishna, "A deep learning based artificial neural network approach for intrusion detection," in *International Conference on Mathematics and Computing*. Springer, 2017, pp. 44–53.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7378–7382.

[5] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[6] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[7] Y. Miao, J. Li, Y. Wang, S. Zhang, and Y. Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 2284–2288.

[8] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[9] Information and I. Computer Science, University of California, *KDD Cup'99 Dataset*, Last modified: October 28, 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[10] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Feature selection using genetic algorithm to improve classification in network intrusion detection system," in *2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, Sept 2017, pp. 46–49.

[11] J. E. Varghese and B. Muniyal, "An investigation of classification algorithms for intrusion detection system — a quantitative approach," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2017, pp. 2045–2051.

[12] Q. Lyu and J. Zhu, "Revisit long short-term memory: An optimization perspective," in *Advances in neural information processing systems workshop on deep Learning and representation Learning*, 2014, pp. 1–9.

[13] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[14] P. Shah, V. Bakarola, and S. Pati, "Image captioning using deep neural architectures," *CoRR*, vol. abs/1801.05568, 2018. [Online]. Available: http://arxiv.org/abs/1801.05568

[15] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *CoRR*, vol. abs/1710.00811, 2017. [Online]. Available: http://arxiv.org/abs/1710.00811

[16] Y. Fu, F. Lou, F. Meng, Z. Tian, H. Zhang, and F. Jiang, "An intelligent network attack detection method based on rnn," in *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, June 2018, pp. 483–489.

[17] M. Alkasassbeh, G. Al-Naymat, N. Hamadneh, I. Obeidat, and M. Almseidin, "Intensive preprocessing of KDD cup 99 for network intrusion classification using machine learning techniques," *CoRR*, vol. abs/1805.10458, 2018. [Online]. Available: http://arxiv.org/abs/1805.10458

[18] G. Meena and R. R. Choudhary, "A review paper on ids classification using kdd 99 and nsl kdd dataset in weka," in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, July 2017, pp. 553–558.

[19] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, Feb 2016, pp. 1–5.

[20] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South African Computer Journal*, vol. 56, no. 1, pp. 136–154, 2015. [Online]. Available: https://journals.co.za/content/comp/56/1/EJC173450

[21] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[22] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[23] B. Athiwaratkun and J. W. Stokes, "Malware classification with lstm and gru language models and a character-level cnn," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2482–2486.

[24] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional LSTM rnns for acoustic modeling in speech recognition," *CoRR*, vol. abs/1606.06871, 2016. [Online]. Available: http://arxiv.org/abs/1606.06871

[25] A. Elsherif, "Automatic intrusion detection system using deep recurrent neural network paradigm," *Journal of Information Security and Cybercrimes Research (JISCR)*, vol. 1, no. 1, 2018.

[26] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential lstm neural networks autoencoders," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, May 2018, pp. 1–4.

[27] T. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *2017 International Conference on Platform Technology and Service (PlatCon)*, Feb 2017, pp. 1–6.

[28] L. Bontemps, V. L. Cao, J. McDermott, and N. Le-Khac, "Collective anomaly detection based on long short term memory recurrent neural network," *CoRR*, vol. abs/1703.09752, 2017. [Online]. Available: http://arxiv.org/abs/1703.09752

[29] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *CoRR*, vol. abs/1803.10769, 2018. [Online]. Available: http://arxiv.org/abs/1803.10769

[30] V. Campos, B. Jou, X. Giró i Nieto, J. Torres, and S. Chang, "Skip RNN: learning to skip state updates in recurrent neural networks," *CoRR*, vol. abs/1708.06834, 2017. [Online]. Available: http://arxiv.org/abs/1708.06834

[31] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," *Proceedings of ACM CSS Workshop on Data Mining Applied to Security Philadelphia PA*, pp. 1–25, 2001. [Online]. Available: http://freeworld.thc.org/root/docs/intrusion_detection/nids/ID-with-Unlabeled-Data-Using-Clustering.pdf

[32] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *International Journal of Security and its Applications*, vol. 9, no. 5, pp. 205–216, 2015.

[33] D. P. Vinchurkar and A. Reshamwala, "A Review of Intrusion Detection System Using Neural Network and Machine Learning Technique," *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 1, no. 2, pp. 54–63, 2012.

[34] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[35] T. Tang, S. A. R. Zaidi, D. McLernon, L. Mhamdi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in sdn-based networks," in *2018 IEEE International Conference on Network Softwarization (NetSoft 2018)*. IEEE, 2018.

[36] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov 1997.

[37] Y. Yu, M. Abadi, P. Barham, E. Brevdo, M. Burrows, A. Davis, J. Dean, S. Ghemawat, T. Harley, P. Hawkins *et al.*, "Dynamic control flow in large-scale machine learning," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 18.

[38] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *Journal of information security*, vol. 7, no. 03, p. 129, 2016.

[39] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[40] J. K. Jaiswal and R. Samikannu, "Application of random forest algorithm on feature subset selection and classification and regression," in *2017 World Congress on Computing and Communication Technologies (WCCCT)*, Feb 2017, pp. 65–68.

[41] S. Amin and A. Singhal, "Identification and classification of neuro-degenerative diseases using feature selection through pca-ld," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, Oct 2017, pp. 578–586.

[42] D. Hong, L. Balzano, and J. A. Fessler, "Asymptotic Performance of PCA for High-Dimensional Heteroscedastic Data," *Journal of Multivariate Analysis*, vol. 167, pp. 435–452, 2017. [Online]. Available: http://arxiv.org/abs/1703.06610

[43] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *CoRR*, vol. abs/1707.06799, 2017. [Online]. Available: http://arxiv.org/abs/1707.06799

[44] Tensorflow.Org, *Tensorflow framework*, Last updated May 25, 2018. [Online]. Available: https://www.tensorflow.org/

Appendix A

Table A.1: All the 41 Features of KDD Cup'99 Dataset

Sr.No	**Feature name**	**Description**	**Type**
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of "wrong" fragments	continuous
9	urgent	number of urgent packets	continuous
10	hot	number of "hot" indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of "compromised" conditions	continuous

14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if “su root” command attempted; 0 otherwise	discrete
16	um_root	number of “root” accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a “guest”login; 0 otherwise	discrete
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	serror_rate	% of connections that have “SYN” errors	continuous
25	rerror_rate	% of connections that have “REJ” errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
29	srv_serror_rate	% of connections that have “SYN” errors	continuous
30	srv_rerror_rate	% of connections that have “REJ” errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous
32	Dst_host_count	count for destination host	continuous
33	Dst_host_srv_count	srv_count for destination host	continuous
34	Dst_host_same_srv_rate	same_srv_rate for destination host	continuous
35	Dst_host_diff_srv_rate	diff_srv_rate for destination host	continuous
36	Dst_host_same_src_port_rate	same_src_port_rate for destination host	continuous
37	Dst_host_srv_diff_host_rate	diff_host_rate for destination host	continuous

38	Dst_host_serror_rate	serror_rate for destination host	continuous
39	Dst_host_srv_serror_rate	srv_serror_rate for destination host	continuous
40	Dst_host_rerror_rate	rerror_rate for destination host	continuous
41	Dst_host_srv_rerror_rate	srv_serror_rate for destination host	continuous

Curriculum Vitae

Name:	Wafaa Anani
Post-Secondary Education and Degrees:	Ajman University Ajman, UAE 1988 - 1995 BSc. Computer Science University of Western Ontario London, ON 2016 Master of Engineering University of Western Ontario London, ON 2017 - Present, Master of Engineering Science
Related Work Experience:	Teaching Assistant The University of Western Ontario 2016 - 2018 Completed Advanced Teaching Program (ATP) The University of Western Ontario 2016

Publications:

- Wafaa Anani and Abdelkader Ouda. *"The Importance of Human Dynamics in the Future User Authentication"*. 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering CCECE, pp. 1-5, 2017.
- Wafaa Anani and Jagath Samarabandu. *"Comparison of Recurrent Neural Network Algorithms for Intrusion Detection Based on Predicting Packet Sequences"*. 2018 IEEE 31st CCECE, pp.1-4, 2018.