A Thesis

entitled

A Study of Effective Framework Combining Sparse Autoencoder Based Feature Transfer Learning and Long Short-Term Memory for Network Intrusion Detection System

by

Zolzaya Kherlenchimeg

Department of Design and Media Technology

Graduate School of Engineering

Iwate University

March 2020

Dedication

To my grandparents, who taught me the most important values of life.

Abstract

The rapid growth of technology uses all over the world, our daily lives and activities for the better in many ways. However, this exponential growth of interconnections has led to also concern network security issues. A vulnerability and potential malicious threat might be due to a bug in applications and ill-managed networks. Therefore, we must address these critical issues of network security such as detect suspicious activities, a countermeasure against intruders and unauthorized access to the existing data. In the last decades, the Intrusion Detection System (IDS) plays a vital role in detecting network attacks. The IDS is a process of monitoring and analyzing the events occurring in a computer system and network to detect signs of security problems.

In general, IDS categorized into misuse-based detection and anomaly-based detection. A misuse-based IDS also known as a signature-based IDS that measures its similarity between input and signatures of known attacks. Therefore, the known attacks can be detected immediately and reliably with a lower false-positive rate. While the misuse-based detection method has disadvantages that it cannot detect unknown attacks and novel attacks. The anomaly-based detection technique is the process of comparing activity the enterprise considers normal against observed activity to identify significant deviations. The advantage of anomaly-based detection techniques is suitable to predict and adapt to unknown attacks. This kind of detection method uses a machine learning approach to create a predictive model by simulating regular activity and known activity, then compare new behaviors with the existing model. However, an anomaly-based IDS usually produces a high percentage of false alarms rate and a low rate of detection rate, it might be effect the efficiency of real-world applications. In an anomaly-based IDS, there are different levels at which an IDS can monitor activities in a network. It faces a large number of features representation of monitored network traffic. The

high dimensionalities of the network traffic give to raise the hypothesis search space and also lead to large classification errors. Therefore, to address those problems, this study focused to improve the accuracy of detecting unknown attacks through the developing an effective framework.

In this study, we propose a network IDS framework for intrusion detection. The proposed framework consists of two stages. The first stage is a feature extraction stage which has two steps: unsupervised pre-training and supervised fine-tuning. The first step is an unsupervised pre-training step that learns the typical patterns of the network traffic using a single-layer Sparse Autoencoder (SAE) which is an effective learning algorithm for reconstructing a new feature presentation of the data through the nonlinear mapping. Consequently, the second step is a supervised fine-tuning step that can extracts the primary features of the network traffic using the preceding optimal parameters in supervised manner while gradually reduce the data dimension. The SAE model determines an approximation to the identity function, so as to output data that is similar to their input data. In other words, the function involves finding the optimal network parameters weight, biases by minimizing the discrepancy between input and its reconstruction data. However, the degree of input features increases the model becomes more complex and has to fit all data. Therefore, to prevent the problem of overfitting, we use the L2 regularization method by augmenting the cost function with the sum of the squared magnitude of all weights in the network. As well as, we regularize the feature extractor model by using a Kullback-Leibler (KL) divergence as a sparsity penalty term which constrains the neurons to be inactive most of the time.

Accordingly, we train a single-layer feature learning SAE model on training set only in unsupervised manner using 5-fold cross-validation, while optimize hyperparameter values of the network. It involves finding the optimal network parameters weight, biases and hyperparameters of cost function. After the network learned optimal values for weights and biases, save the network parameters. Once selecting proper hyperparameter, re-train the feature extractor SAE model using these optimal hyperparameter on the training set with a label. Finally, the feature extractor SAE model can extract the new feature representations which represent the source data. In the next stage, train a Long Short-Term Memory model to identify the network traffic as being either normal or attack using the extracted new feature representations dataset. We

apply the 10-fold cross-validation method to validate the results on the LSTM model to prevent overfitting issue. In final, we evaluate the effectiveness of the proposed IDS framework on the benchmark NSL-KDD dataset. The experimental result shows that the proposed framework performs better than previous studies which proves the effectiveness of our framework. Furthermore, the result confirmed that our feature extractor SAE model significantly effected to improve the performance of this work.

This thesis was aimed to develop an effective framework combining a single-layer Sparse Autoencoder (SAE) based feature transfer learning and Long Short-Term Memory (LSTM). Initially, the feature extractor SAE model which proposed to extract the most relevant features for use in representing the data. In the following, the LSTM method proposed for classifying network traffic either a normal or an attack. The result of the proposed framework detected network attack with high accuracy and it outperformed other similar studies.

Acknowledgements

I would like to take this opportunity to acknowledge the people who made this thesis possible.

First and foremost, I want to express my special gratitude to my supervisor, Associate Professor Naoshi Nakaya for his support, patient guidance, encouragement, and kind advice during the long process of this thesis.

Besides, I would like to extend my special thanks to the thesis committee members, Professor Kouichi Konno, Professor Tadahiro Fujimoto and Associate Professor Takuya Akashi for your insightful questions and suggestions to make my work stimulating. I am particularly grateful for the assistance given by Ms. Hikaru Kaketa, since my first day of a doctoral student.

I gratefully acknowledge the funding received towards my Ph.D from the Higher Engineering Education Development (M-JEED) project, especially my gratitude to Professor Enkhbayar Altantsetseg for his encouragement. I am also sincerely thankful to Dr. Batjargal Sosorbaram who has been supporting us while living and studying in Japan.

Last but not least, I would like to thank my family, especially my lovely son, Bayasgalan and my friends. Without their support and encouragement, I would not have been able to complete this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The vast majority of our daily lives and activities are availability for the use of the Internet. In recent years, the growth in network traffic is being driven by increased mobile devices, Internet of Things (IoT) devices and a continued increase in average data volume per device. According to the recent report of Cisco Visual Networking Index (VNI) forecast [3], the number of global IP traffic will grow by 396 exabytes (EB) per month IP traffic, it will be reached threefold to 2022. However, this exponential growth of the Internet interconnections has lead to significant growth of cyber-threat incidents. In particular, network vulnerability is a weak spot in the network that might be exploited by a security threat. According to the Symantec Information Security Threat report published in February 2019 [4], 4800 websites per month compromised with formjacking code which uses of malicious code to steal credit card details and other information from payment forms. Therefore, many researchers to address these critical issues of network security, such as detecting suspicious activities and need countermeasures against a diverse range of threats. As a shown in Figure 1.1, the Hackmageddon Information Security Timelines and Statistics Website has reported the top 10 attack distribution of the last year.

In terms of the attacker's purpose, cybersecurity risks can be broadly divided into two types: active and passive attacks. An active attack attempts to alter system resources or effect their operations. Active attack involve some

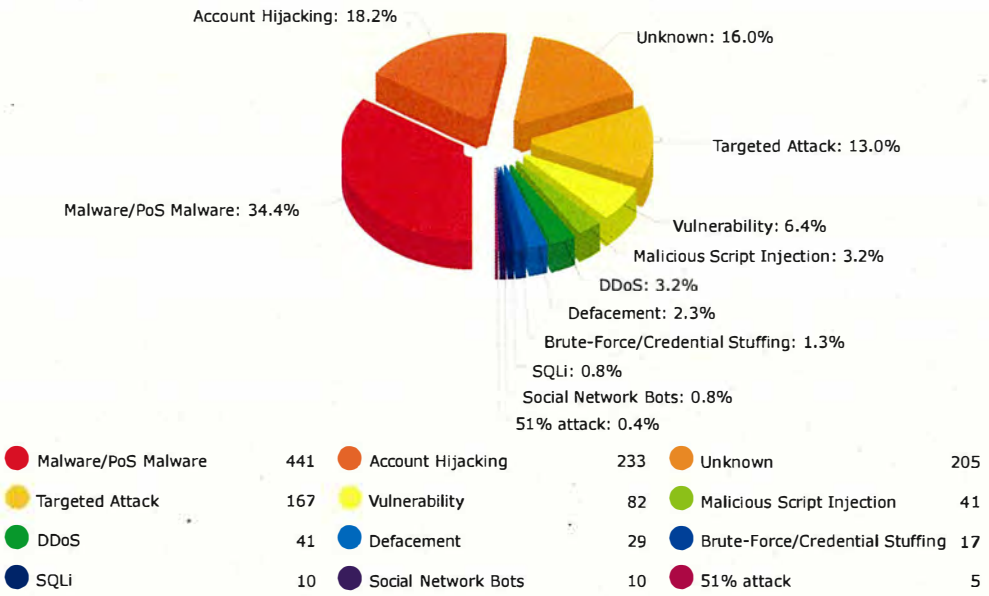

Figure 1.1: Top 10 distribution of attacks (2018) [1]

modification of the data stream or creation of false statement. So it compromises integrity or availability. In contrast, a passive attack attempts to learn or make use of information from the system but does not affect system resources. Passive attacks are in the nature of eavesdropping on or monitoring of transmission. So it compromises confidentiality. In order to protect network infrastructure against potential malicious threats, the growing effort collaboration between research communities and cybersecurity professionals from industry, academia, government agencies.

There are numerous conventional techniques to cyber defense, for example firewalls, access control, antivirus software and intrusion detection system. However, these type of defense system have few limitations, particularly it depends on their design and implementation of software and network infrastructure. Because patches have been developed to protect the systems, but attackers continuously exploit another vulnerability.

The goal of the opponent is to obtain information is being transmitted. Therefore, cybersecurity concerns with the understanding of surrounding issues of diverse cyber attacks that preserve confidentiality, integrity, and availability of any digital and information technologies. Because, a poorly im-

plemented information security system can in itself become a source of risk, however, so organizations must ensure that their information security systems address the CIA triad:

- **Confidentiality** (C) is the term used to protect the information from access by unauthorized individuals or parties.
- **Integrity** (I) is the term used to prevent any alter or modify in an unauthorized manner.
- **Availability** (A) is the term used to assure that the systems responsible for delivering, storing and processing information are accessible when needed and by those who need them.

The CIA triad is the basis of information security. In order to provide the CIA triad, diverse conventional defense strategies are building for information security. Over the last decades, Intrusion Detection System (IDS) has been playing a vital role in detecting network attacks. The IDS is a hardware appliance or software application that an intelligently monitors activities that occur in a computing resource, network traffic, computer usage, and to analyze the events, to generate the reactions. Typically, IDSs categorize into misuse-based detection and anomaly-based detection. A misuse-based IDS also known as a signature-based IDS that measures its similarity between the input and signatures or pattern of known attacks. Thus the known attacks can be detected immediately and reliably with a lower false-positive rate. While the misuse-based detection method has disadvantages that it cannot detect unknown attacks and novel attacks. Snort [5] is a popular signature-based IDS and network specialist needs to update attack signature database into IDS. An anomaly-based detection technique is designed to detect patterns that deviate from established normal usage patterns that can be flagged as an attack. The advantage of anomaly-based detection techniques is suitable to predict and adopt to unknown attacks. This detection method uses machine learning approach to create a predictive model simulating regular activity, and then compares new behavior with the existing model. Thus, most researchers emphasize the anomaly-based detection method which can give better performance in credit card fraud detection, medical diagnosis, fault detection [6]. However, anomaly-based IDSs usually produce a high percentage of false alarms, it might reduce the efficiency of real-world applications.

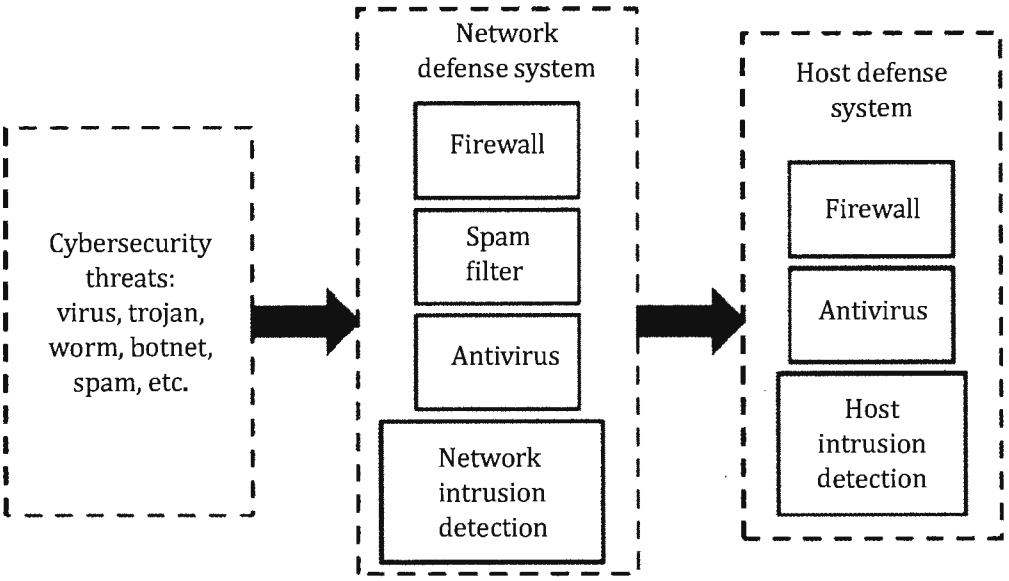

Figure 1.2: Conventional cybersecurity system [2]

As shown in Figure 1.2, conventional cybersecurity systems address various cybersecurity threats, including viruses, trojans, worms, spam, and botnets. These cybersecurity systems run against malicious threats at two classes: a host-based intrusion detection system (HIDS) and network-based intrusion detection system (NIDS). HIDS is installed on each client of the network and can monitor particular clients only. In contrast to HIDS, a NIDS is placed in a network to detect attack on the hosts of that network.

1.2 Problem Statement

Cybersecurity defense strategy is a collection of policies, techniques, software, and hardware, that can protect an application, network, host, information from attacks. There are various conventional techniques to cyber defense. These type of techniques has few limitations, particularly it depends on their design and implementation of software and network infrastructure.

Due to the availability of large amounts of data, machine learning algorithms is successfully exploited in network security problems, for instance the literature [7] review to focusing on attacks related to spam detection, malware analysis, and intrusion detection. In existing studies, a variety of machine learning algorithms were used to develop a NIDS.

In anomaly detection, there are different levels at IDS that monitors activities in a network. It faces a large number of features representing the monitored network traffics. The high dimensionality of the network traffic data gives a large hypothesis search space, and also can lead to large classification errors. Therefore, in order to build an effective IDS framework, this study pay attention to select the most relevant features for use in representing the data using nonlinear mapping.

1.3 Contributions

Considerable researches have been done to avoid cyber security attacks. Over the years, a study of single-layer [8] [9] neural network for unsupervised feature learning gained increasing attention. In this thesis, we demonstrate the development of a novel effective framework for network intrusion detection system. The proposed framework consists of two stages. In the first stage, a single-layer Sparse Autoencoder (SAE) is utilized to discover the effective features of the input data, and gradually reduce the data dimension. In the second stage, train a Long Short-Term Memory (LSTM) model using a preceding parameters to identify type of network traffic. The main contributions of this thesis are summarized as follows:

- We aim to present an effective framework combining a single-layer Sparse Autoencoder and Long Short-Term Memory for network intrusion detection system. By taking an unsupervised learning method to learn useful features from raw data for improving accuracy of the predictive model while reducing the data dimension.
- The result proved that our SAE feature extractor model significantly effected to improve the performance of this work.
- Our experiment demonstrates that the SAE-LSTM has a greater performance than other state-of-the-art and anomaly-based detection methods on the benchmark NSL-KDD dataset.

1.4 Thesis outline

The overall organization of the thesis is presented in this section.

Chapter 2 discusses the most illustrative works of intrusion detection system applied to the benchmark NSL-KDD dataset, along with a general discussion about works on machine learning and deep learning approaches in cybersecurity issues.

Chapter 3 provides a fundamental issue of understanding the detection aspect that was discussed in the Introduction. Then provides a comprehensive discussion on deep learning approaches.

Chapter 4 introduces the first attempt to design a deep learning approach that combined sparse autoencoder with recurrent neural networks for the intrusion detection system. We evaluate the performance of the proposed method on the NSL-KDD dataset.

Chapter 5 presents an effective framework combining the SAE based feature transfer learning and LSTM for NIDS. We evaluated the performance of the framework on testing dataset. And then, the results of proposed framework compared with similar previous studies.

Chapter 6 concludes the development of the proposed framework and their results; as well as insights to overcome the limitations of our work along with the enhancements.

Chapter 2

Literature review

This section presents the most illustrative works of intrusion detection system applied to the benchmark NSL-KDD dataset, along with a general discussion about works on machine learning and deep learning approaches in cybersecurity issues.

2.1 Challenges

Most commercial products contain signature-based detection techniques [2]. Although many methods and systems have been developed by the research community, there are still a number of open research issues and challenges:

- Network based IDS monitor the whole network, therefore vulnerable to the same attacks the network's hosts are.
- An anomaly-based NIDSs usually produce a high percentage of false alarm. But, totally mitigating the false alarm is not possible.
- Developing a suitable method for extracting the features for each category of attack is another important task.
- Identifying a best classifier that is non-associated and unbiased to build an effective approach for anomaly detection is another challenge.

2.2 Related Works

There are a large number of related studies using either the KDD-Cup 99 or DARPA 1999 dataset to validate the development of IDSs. In this section, we review the literature that used to apply machine learning and deep learning techniques for intrusion detection.

In existing studies, a variety of machine learning algorithms were used to develop a network-based IDS, for instance, Artificial Neural Network (ANN) [10], k-Nearest Neighbor (k-NN) [11] [12], Support Vector Machines (SVM) [9] [13]. Furthermore, researcher community have been attracting their attention to a deep learning approaches including Deep Neural Network [14], Self-Taught Learning [15], Stacked Autoencoder [16] [17], Recurrent Neural Network [18]-[19], Convolutional Neural Network [20] [21] for intrusion detection. The literature [22] has demonstrated a simple feature learning framework that incorporates an unsupervised learning algorithm.

Most of these studies suggested extracting a relevant pattern from network traffic, to attain further improvement in an overall accuracy of the system. As well as, we emphasize on the researches which used to evaluate their performance on the NSL-KDD dataset. A work proposed by Tang *et al.* [14], deep neural network (DNN) model for flow-based anomaly detection in the Software-Defined Network (SDN) environment. The proposed model is compared with a state-of-the-art algorithms through the use of accuracy at 75.75%, which is utilizing basic 6 features from the NSL-KDD dataset.

Ingre *et al.* [10] has proposed IDS using Artificial Neural Network (ANN) on the NSL-KDD dataset. The work utilized Levenberg-Marquardt (LM) and BFGS quasi-Newton Backpropagation algorithm for training in binary and 5-class classification. They reduced feature set of 29 by removing least usable features from the training and testing set, and then obtained 81.2% of accuracy for binary classification.

Within a development of deep learning, the representation learning approach [23] allows a system can automatically extract features from raw data. For instance, Javaid *et al.* [15] further introduced a Self-Taught Learning (STL) based on autoencoder with softmax regression to implement a NIDS. The proposed method developed two different models in binary and multiclass classification and evaluated on the NSL-KDD dataset. STL achieved 88.39%

accuracy rate for binary classification and outperformed the previous studies results.

In the paper [24], we demonstrated a NIDS based on sparse autoencoder (SAE) technique to reduce the dimensionality of network traffic, followed by the RNN as a classifier. The performance of the proposed model was evaluated on the NSL-KDD dataset, and then the result compared with prior studies at 80.0% of accuracy rate.

Yousefi-Azar *et al.* [25] proposed an unsupervised feature learning approach for malware classification and network-based anomaly detection using deep autoencoder (DAEs). The authors provided 10-dimensions conceptual space for a latent layer of DAEs in both malware and intrusion detection tasks. The output of latent layer was fed into various classification methods such as SVM, k-NN and Gaussian Naive Bayes. The experimental results proven that the proposed DAEs is a more efficient in dimension reduction that compared with result of the original features of the NSL-KDD dataset.

Similarly, Li *et al.* [26] built an autoencoder to reduce the dimensionality of the KDDCUP'99 dataset, followed by deep belief network (DNB) classifier that achieved an accuracy of 92.1% with a FPR of 1.58%.

As well as, a few researchers have been proposed a potential IDSs based on individual deep learning approaches including recurrent neural network (RNN) and convolutional neural network (CNN). The paper introduced by Yin *et al.* [27], a deep learning approach for intrusion detection using recurrent neural network (RNN-IDS). This work was obtained good result, especially under the task of multiclass classification with 81.29% of accuracy on the NSL-KDD dataset. Kim *et al.* [18] implemented a similar test of IDS classifier using LSTM-RNN. The authors considered a problem of imbalance on the KDDCup'99 dataset, and then generated a new training set by extracting 300 instances from each attack types and 1000 normal instances. The proposed IDS was gained 98.88% of detection rate and 10.04% of false alarm rate. Moreover, Li *et al.* [20] proposed an intrusion detection system using convolutional neural networks (CNN) that adopts novel representation learning methods of graphic conversion. The method of transforming standard NSL-KDD dataset data form into 8*8 gray-scale images is introduced. They used the ReSNet50 and GoogLeNet network as CNN models.

2.3 Conclusion

A number of surveys and review articles have focused on intrusion detection technologies. To address the issue, we design to create an effective novel framework for intrusion detection using sparse autoencoder and LSTM algorithms. Then, we try to build an effective framework to evaluate on the benchmark NSL-KDD dataset and compare it to the recent studies results.

Chapter 3

Background

This chapter starts with a fundamental issue of understanding the detection aspect that was discussed in the Introduction. Then provides a comprehensive discussion on deep learning approaches.

3.1 Intrusion detection system

An intrusion detection system (IDS) analyze and monitor network traffic for signs that indicate attackers are using a known cyber threat to infiltrate or steal data from the network. An intruder to a system is very likely to exhibit a pattern of behavior different from the normal behavior of a legitimate user. The IDS types range in scope from single computers to large networks [28].

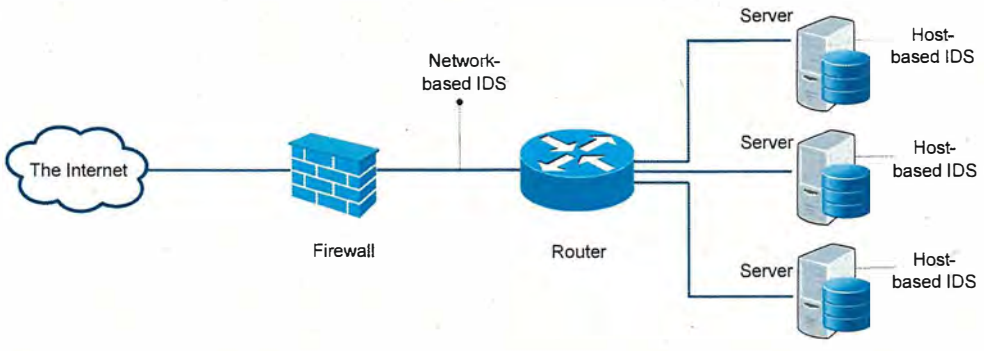

Figure 3.1: Type of IDS architecture

Intrusion detection systems are usually passive devices that are not configured to automatically take any punitive action against network traffic that appears to be malicious. Intrusion detection identifies that an intrusion is taking place and informs an administrator who must take appropriate action. It is categorized based on their data source into two main categories: network based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS). Figure 3.1 shows that type of IDS architecture.

The NIDSs are strategically placed on system to monitor all the network traffic. It is widely deployed in modern enterprise network. A NIDS reads all inbound network traffic and matches the traffic that is passed on the network to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator. NIDS can be also combined with other technologies to increase detection and prediction rates. Artificial Neural Network (ANN) based IDS are capable of analyzing huge volumes of data, in a smart way, due to the self-organizing structure that allows IDS to more efficiently recognize intrusion patterns [29]. Neural networks assist IDS in predicting attacks by learning from mistakes. IDS help develop an early warning system, based on two layers. The first layer accepts single values, while the second layer takes the first's layers output as input. The cycle repeats and allows the system to automatically recognize new unforeseen patterns in the network [30].

A host based IDSs are installed on specific host, analyzing traffic and logging intrusion behavior, then it will log the activity. A HIDS monitors the inbound and outbound network traffic from the device only and will alert the user or administrator if suspicious activity is detected. It takes a capture of existing system call files [31] and matches it to the previous capture. If the critical system call files were modified or deleted, an alert is sent to the administrator to investigate that suspicious activities.

Traditionally, the researchers study intrusion detection approaches from two major perspectives: anomaly and misuse detection that was discussed in the Chapter 1. There are many studies mentioned that an anomaly-based network intrusion detection plays a vital role in protecting networks against malicious activities. Based on these observations, we advocate that to develop NIDS can be also combined with deep learning technologies to increase detection and prediction rates.

3.2 Methodology

Deep learning is a sub-field of machine learning that uses artificial neural networks (ANNs) containing two or more hidden layers to approximate some function $f(\cdot)$, where $f(\cdot)$ can be used to map input data to new representations or make predictions. The ANNs inspired by the biological neural network is a set of interconnected neurons or nodes, where connections are weighted and each neuron transforms its input into a single output by applying a non-linear activation function to the sum of its weighted inputs.

3.2.1 Autoencoder

An autoencoder is an unsupervised learning algorithm that mainly used as feature extraction or a dimensionality reduction. Hinton [32] *et al.* demonstrated the potential of autoencoders for trying to learn an approximation to the identity function, so as to output vector $\hat{X}$ that is similar to the input vector X. Basically, an autoencoder is simply a multilayer feedforward neural network trained to represent the input with backpropagation.

The autoencoder is a symmetric neural network structurally defined by three layers: an input layer, hidden layers, and an output layer. The input and output layer has a same number of N units and the hidden layer contains K units. Figure 3.2 shows the schematically layout of an autoencoder (AE) comprises encoder and decoder. The aims of encoder compress the input vector $X = (x_1, x_2, x_3....., x_n)$ into a low dimension representation $H = (h_1, h_2, h_3....., h_k)$. It can be represented by an encoding function as illustrated in Eq. 3.1. This compressed representation is decompressed by the decoder to reconstruct the output vector $\hat{X} = (\hat{x_1}, \hat{x_2}, \hat{x_3}....., \hat{x_n})$. The decoding function as expressed in Eq. 3.2. This way, the latent space forms a bottleneck, which forces the autoencoder to learn an effective compression of the data.

$$h = f\left(\sum_{i=1}^{n} W x_i + b\right) \tag{3.1}$$

$$\hat{x} = f\left(\sum_{i=1}^{k} W' h_i + b'\right) \tag{3.2}$$

where weight matrices $W \in \Re^{k \times n}$ and $W' \in \Re^{n \times k}$ of the encoder and decoder network, respectively. $b \in \Re^k$ and $b' \in \Re^n$ are bias vectors. The activation function f(·) takes the input of a neuron z and outputs signal y. $f(z)$ denotes a sigmoid activation function. As seen in Eq 3.3, the sigmoid activation function is one of the most frequently used non-linear activation functions for feedforward neural networks [33] that normalize the output of each neuron to output values bound between 0 and 1 nonlinearly, which is fully differentiable:

$$f(z) = \frac{1}{1 + \exp^{-z}} \tag{3.3}$$

where $z = b + \sum_{i=1}^{n} x_i w_i$ Parameters W, b are optimized using back propagation, by minimizing the cost function $J_{error}(W, b)$ for training set n is described in Eq. 3.4. The first term is an average of sum-of-square errors between x_i input vector and $\hat{x_i}$ reconstructed vector.

$$J_{error}(W, b) = \frac{1}{2n} \sum_{i=1}^{n} ||\hat{x_i} - x_i||^2 + \frac{\lambda}{2} (\sum_{k,n} W^2 + \sum_{n,k} W'^2) \tag{3.4}$$

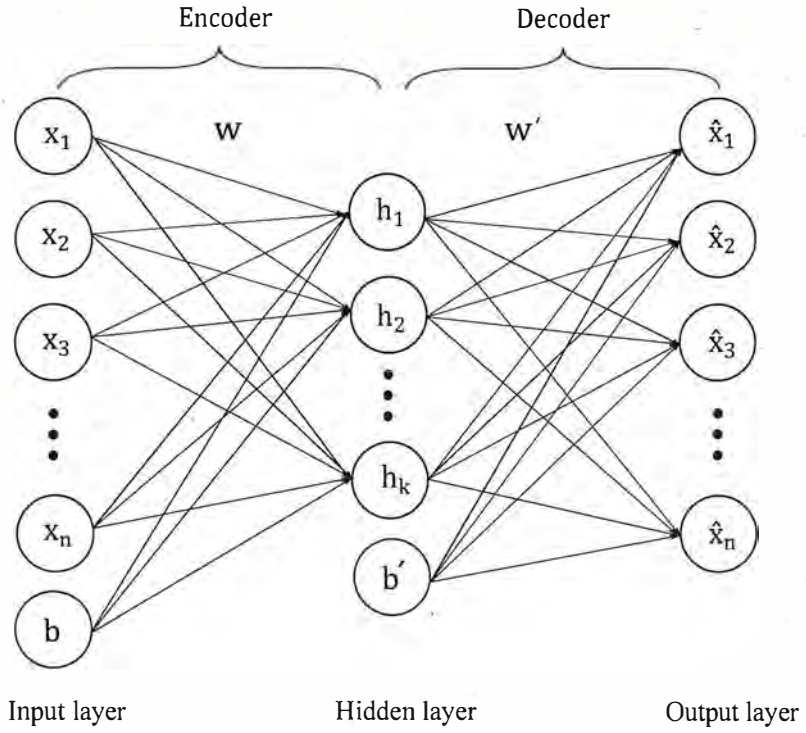

Figure 3.2: Structure of pre-training sparse autoencoder

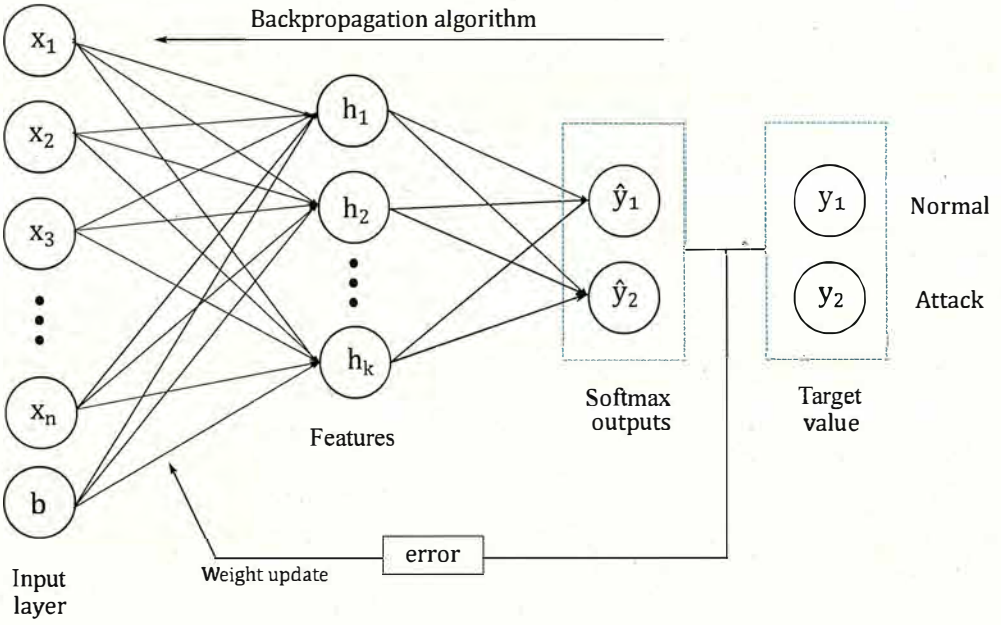

Figure 3.3: Structure of fine-tuning sparse autoencoder

The second term indicates an L2 regularization (a weight decay term) penalizing term by augmenting the cost function with the sum of the squared magnitude of all weights in the neural network and it prevents problem of overfitting. The λ (lambda) is called as a regularization parameter which determines how much to penalizes the weights.

3.2.2 Sparse Autoencoder

There are numerous ways of defining a simpler representation. The most common include lower is a sparse representations [32] [34]. Sparse presentations embed the dataset into a representation whose entries are mostly zeroes for most inputs. The use of sparse representations typically requires increasing the dimensionality of the representation, so that the representation becoming mostly zeroes does not discard too much information. In order for the sparseness of hidden units, regularize autoencoder by using a sparsity penalty term which constrains the neurons to be inactive most of the time [35]. Sparsity is a useful constraint when the number of hidden units is large. Therefore, a sparsity penalty term is added to the Eq. 3.4, and the new objective functions

are given as follows:

$$J_{sparse}(W,b) = J_{error}(W,b) + \beta \sum_{j=1}^{n} KL(\rho \,||\, \hat{\rho}_j) \tag{3.5}$$

The sparsity penalty term is regularized by Kullback-Leibler (KL) divergence [36] which is a measure of the difference between two probability distributions. The KL divergence between two probability distributions ρ and $\hat{\rho}_j$ is defined as:

$$KL\left(\rho \,||\, \hat{\rho}_j\right) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{3.6}$$

where $\hat{\rho}_j$ (rho) is an average activation of hidden unit j over the training set n, which can be formulated

$$\hat{\rho}_j = \frac{1}{n} \sum_{i=1}^{n} h_j(x_i) \tag{3.7}$$

ρ is predefined mean activation target $\rho \in \{0, 1\}$, and the most values of hidden units are much smaller than given a small ρ. This addition of sparsity constraints can lead the learned hidden representation to be a sparse representation. Therefore, the variant of autoencoder is named sparse autoencoder. In here, the β (beta) controls the weights of the sparsity penalty term.

The optimizing function defines how we are going to update our parameters (weights and biases) in order to reduce the reconstruction error. The single-layer SAE model is trained using stochastic gradient descent optimization algorithm and the network parameters W, b are updated through the back-propagation algorithm based on the minimizing the cost function $J_{sparse}(W,b)$. The model can extract meaningful features from the input data, instead of simply converting the input vector.

3.2.3 Recurrent Neural Network

The recurrent neural network (RNN) was first developed in the 1980s [37]. It is a most powerful neural network with cyclic connections which are widely used in machine translation, speech synthesis, speech recognition as well as for processing a sequence of values x^1, ..., x^n [38] [39]. The structure consists of an input layer, one or more hidden layers and output layer.

Unlike feedforward neural networks, RNNs add a neural node based on the architecture of common neural networks, and they memorize the previous

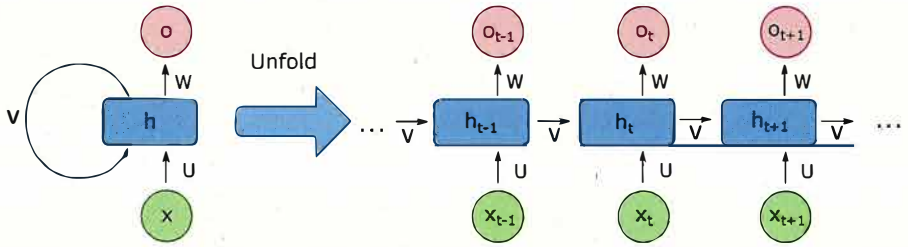

Figure 3.4: Recurrent neural network

state of the neural networks as shown in Figure 3.4. The computational graph to compute the training loss of a recurrent network that maps an input sequence of x values to a corresponding sequence of output o values. The RNN has input to hidden connections parametrized by a weight matrix U , hidden-to-hidden recurrent connections parametrized by a weight matrix W, and hidden-to-output connections parametrized by a weight matrix V.

The most effective sequence models used in practical applications are called gated RNNs. These include the long short-term memory and networks based on the gated recurrent unit. Leaky units did this with connection weights that were either manually chosen constants or were parameters. Gated RNNs generalize this to connection weights that may change at each time step. The long short-term memory (LSTM) [40] have been introduced one of the special form of the RNN which precisely designed to escape the long term dependency issue of recurrent networks. LSTM network further add a component called forget fate, and LSTMs can effectively learn temporal features from a long sequence. LSTM cell called repeating module has four neural network layers interacting with each others as illustrated in Figure 3.5.

$$f_t = \sigma\left(W_f x_t + W_f h_{t-1} + b_f\right) \tag{3.8}$$

$$i_t = \sigma\left(W_i x_t + W_i h_{t-1} + b_i\right) \tag{3.9}$$

$$c_t = f_t * c_{t-1} + i_t * \tanh\left(W_c x_t + W_c h_{t-1} + b_c\right) \tag{3.10}$$

$$o_t = \sigma\left(W_o x_t + W_o h_{t-1} + b_o\right) \tag{3.11}$$

$$h_t = o_t * \tanh(c_t) \tag{3.12}$$

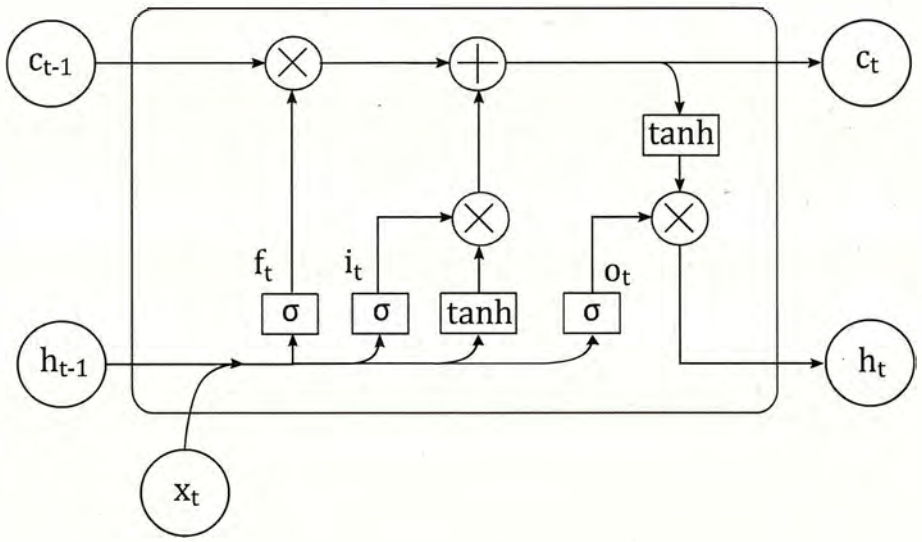

Figure 3.5: Structure of the Long Short-Term Memory cell

σ is the logistic sigmoid function, and f_t, i_t, c_t, o_t are denoted forget gate, input gate, cell state gate, and output gate, respectively. As seen in Eq. 3.8, the forget gate f_t that provides a forgetting coefficient by looking at the input layer x_t and previous hidden layer h_{t-1} for previous cell state c_{t-1}. The output comes out between 0 and 1, and controls the information to forget or pass through from previous cell state c_{t-1} to current cell state c_t. The input gate also considers input layer x_t and previous hidden layer h_{t-1}. It decides which information should be updated in cell state c_t in referred . As described in Eq. 3.10, output of the cell gate c_t is element-wise summation to merge previous information and the current input. In finally, the output gate o_t controls which information should be going to the next hidden state h_t.

Chapter 4

Deep learning based NIDS

In this chapter, we present the first attempt to design a deep learning approach that combined sparse autoencoder with recurrent neural networks for the intrusion detection system.

4.1 Proposed method

In this study, we have proposed the deep learning approach where Sparse Autoencoder (SAE) and Recurrent Neural Network (RNN) are combined for the network intrusion detection. We evaluate the proposed method based on different performance metrics by applying it to the NSL-KDD dataset. Figure 4.1 shows the flow diagram of the overall architecture of the proposed method. During the dimensionality reduction step, the pre-processed data file which has 122 features is loaded into DataFrame using pandas Python package. The DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. An input, hidden, and output layer of the autoencoder feature extraction model contains 122, 60, 122 nodes, respectively. After building the autoencoder model, the data dimension is reduced into 60 features. Then these features are written into csv file. In the RNN classifier, the low dimensional representation data file with 60 features is loaded into the dataframe as the input layer, and finally, we attached Softmax function as a classifier, to classify the data is normal or abnormal.

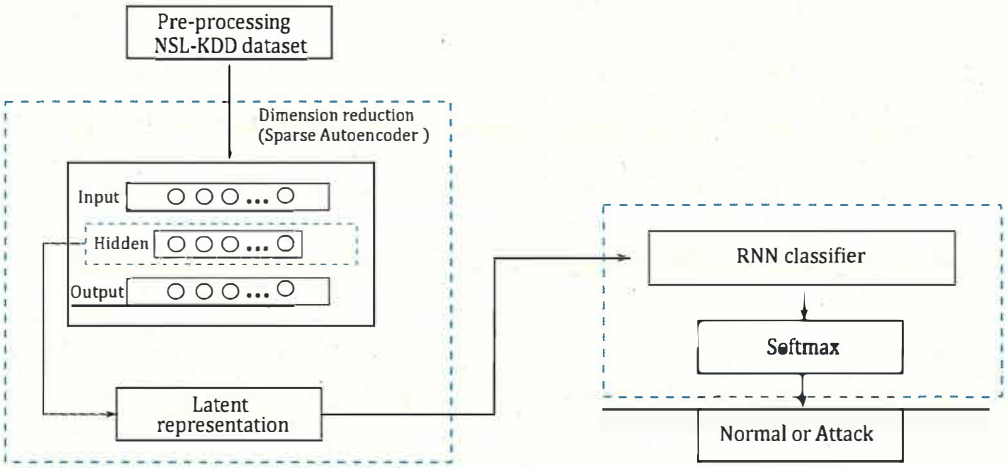

Figure 4.1: Overall architecture of the proposed IDS framework

4.2 The NSL-KDD Dataset

The public benchmark datasets enable researchers to develop the models and compare the performances of the models with the previous research to process the data which are similar to the benchmark dataset. In this study, we use Knowledge Discovery and Dissemination (KDD) 1999 dataset, which is the most widely used dataset for intrusion detection [41]. The dataset was built based on the data captured in DARPA-98, which was developed specifically for Network Intrusion Detection System (NIDS) research is preferred by most researchers. However, there were a large number of redundant records that biased the dataset. Therefore, a work [42] proposed a new dataset named NSL-KDD, which is commonly used in recent studies [43]. The dataset includes the $KDDTrain^+$ as a training set, $KDDTest^+$ as a testing set. As illustrated in Table 4.1, the training set has 125973 network records which contain normal data and 22 types of attack data. The testing set has 22544 network records which contain normal data and 37 types of attack data.

Each record has 41 features categorized into four groups [44] as presented in Table 4.2:

- Basic features: Basic features can be derived directly from packet headers without inspecting the payload. This category contains features 1–9.

Table 4.1: Overview on NSL-KDD

	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain$^+$	125973	67343	45927	11656	52	995
KDDTest$^+$	22544	9711	7458	2421	67	2887

- Content-based features: Domain knowledge is used to access the payload of the original TCP packets. For instance, the R2L and U2R attacks are embedded in the payloads and normally involve only a single connection. To detect these kinds of attacks, one needs some features to be able to look for suspicious behavior in the payloads. This includes features such as number of failed login attempts. This category contains features 10–22.

- Time-based features: These features hold the analysis of the traffic input over a two-second window and contains information like how many connections it attempted to make to the same host. This category contains features 23–31.

- Host-based features: These features are similar with previous group, instead of analyzing over a two seconds window, how many requests made to the same host over x-number of connections. This category contains features 32–41.

For instance, Figure 4.2 shows a distribution of the features in randomly selected 26^{th} line of traffic records of the training set.

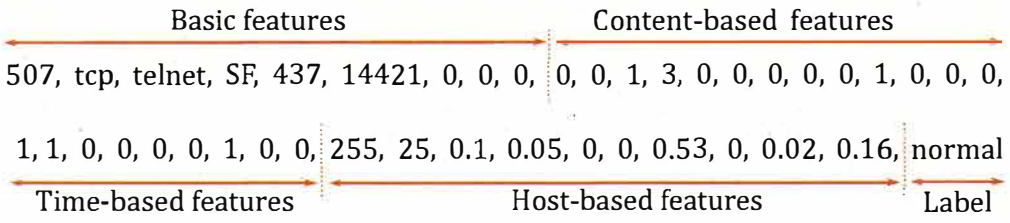

Figure 4.2: The distribution of the features of traffic records in NSL-KDD

Table 4.2: List of features of NSL-KDD dataset

No.	Feature	Description	Type
1	duration	Length of the connection in seconds	continuous
2	protocol type	Type of protocol: *tcp, udp, icmp*	symbolic
3	service	Network service on the destination, *http, telnet, etc.*	symbolic
4	flag	Normal or error status of the connection: *SF, S0, S1, etc.*	symbolic
5	src_bytes	Number of data bytes from source to destination	continuous
6	dst_bytes	Number of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port; 0 otherwise	symbolic
8	wrong_fragment	Number of bad checksum packets in a connection	continuous
9	urgent	Number of urgent packets	continuous
10	hot	Number of "hot" indicators: *entering a system directory, etc.*	continuous
11	num_failed_logins	Number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	symbolic
13	num_compromised	Number of compromised conditions	continuous

Table 4.2 – continued from previous page

No.	Feature	Description	Type
14	root_shell	1 if root shell is obtained; 0 otherwise	continuous
15	su_attempted	1 if "su root" command attempted; 0 otherwise	continuous
16	num_root	Number of "root" accesses	continuous
17	num_file_creation	Number of file creation operations	continuous
18	num_shells	Number of shell prompts	continuous
19	num_access_file	Number of operations on access control files	continuous
20	num_outbound_cmds	Number of outbound commands in a ftp session	continuous
21	is_host_login	1 if login is the "root/admin group"; 0 otherwise	symbolic
22	is_guest_login	1 if login is the "guest" login; 0 otherwise	continuous
23	count	Number of connections to the same host as the current connection in the past 2 seconds	continuous
24	srv_count	Number of connections to the same service as the current connection in the past 2 seconds	continuous
25	serror_rate	% of connections that have "SYN" errors in the *count* feature	continuous
26	rerror_rate	% of connections that have "REJ" errors in the *count* feature	continuous

Table 4.2 – continued from previous page

No.	Feature	Description	Type
27	srv_serror_rate	% of connections that have "SYN" errors in the *srv_count* feature	continuous
28	srv_rerror_rate	% of connections that have "REJ" errors in the *srv_count* feature	continuous
29	same_srv_rate	% of connections to the same service among the connections in the *count* feature	continuous
30	diff_srv_rate	% of connections to different services among the connections in the *count* feature	continuous
31	srv_diff_host rate	% of connections to different hosts among the connections in the *srv_count* feature	continuous
32	dst_host_count	Number of connections to the same destination IP address	continuous
33	dst_host_srv_count	Number of connections to the same destination port number	symbolic
34	dst_host_same_srv_rate	% of connections that same service among the connections in the *dst_host_count* feature	continuous
35	dst_host_diff_srv_rate	% of connections that different service among the connections in the *dst_host_count* feature	continuous

Table 4.2 – continued from previous page

No.	Feature	Description	Type
36	dst_host_same_src_port_rate	% of connections that same source port among the connections in the *dst_host_srv_count* feature	continuous
37	dst_host_srv_diff_host_rate	% of connections that different destination hosts among the connections in the *dst_host_srv_count* feature	continuous
38	dst_host_serror_rate	% of connections that have "SYN" errors in the *dst_host_count* feature	continuous
39	dst_host_srv_serror_rate	% of connections that have "SYN" errors in the *dst_host_srv_count* feature	continuous
40	dst_host_rerror_rate	% of connections that have "REJ" errors in the *dst_host_count* feature	continuous
41	dst_host_srv_rerror_rate	% of connections that have "REJ" errors in the *dst_host_srv_count* feature	continuous
42	label	Network traffic type	symbolic

The NSL-KDD dataset is labeled either normal or an attack, with exactly one specific attack type. There are four major categories of attacks labeled in NSL-KDD: Denial of Service attack, Probing attack, Users-to-Root attack, and Remote-to-Local attack.

- Denial of Service (DoS): Denial of service is an attack category, which exhausts the victim's assets, thereby making it unable to handle le-

gitimate requests. For examples, "neptune", "smurf", "ping of death (pod)", etc.

- Remote-to-Local (R2L): The attackers access the targeted system or network from the remote machine and try to gain the local access of the victim machine. For examples, "guess password", "spy", "ftp write", etc.

- User-to-Root (U2R): The attacker enters into the local system by using the authorized credentials of the victim user and tries to exploit the vulnerabilities to gain the administrator privileges. For examples, "buffer overflow", "rootkit", "load module", etc.

- Probe: Objective of surveillance and other probing attacks is to gain information about the remote victim. An example of probing attacks are "nmap", "portsweep", "satan", etc.

The details of each category are described in Table 4.3. These 17 unseen network attacks are written in bold, it does not include in the training set. This makes the task more realistic.

Table 4.3: Attack types and categories

Category	Attacks in training set	Attacks in testing set
DoS	back, land, neptune, pod, smurf, teardrop	back, land, neptune, pod, smurf, teardrop, **mailbomb, processtable, udpstorm, apache2, worm**
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, warezmaster, **xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named**
U2R	buffer-overflow, perl, loadmodule, rootkit	buffer-overflow, loadmodule, perl, rootkit, **sqlattack, xterm, ps**
Probe	ipsweep, portsweep, nmap, satan	ipsweep, portsweep, nmap, satan, mscan, **saint**

4.2.1 Data Preprocessing

The 41 features of the NSL-KDD dataset, it consists of 38 numerical features and 3 symbolic features. To accelerate the training of neural network, input vector have to be a numerical value. Therefore, we first convert symbolic features into a numerical value using the one-hot (*1-to-n*) encoding. There are 3 features named "protocol_type", "service", and "flag", where "protocol_type" has 3 different symbolic values, "service" has 70 different symbolic values and "flag" has 11 different symbolic values as shown in Table 4.4. For example, "protocol_type" has 3 symbolic values: tcp, udp, icmp and it presented as 0,0,1, 0,1,0, and 1,0,0, respectively. In the same manner, we applied the one-hot encoding to other 2 features, it totally produced 84 new feature values. A collapse of the possible values for the symbolic features can be seen in the

Table. 4.4.

Table 4.4: List of features with symbolic values

No.	Features	Values
1	Protocol Type (2)	tcp, udp, icmp
2	Service (3)	other, link, netbios_ssn, smtp, netstat, ctf, ntp_u, harvest, efs, klogin, systat, exec, nntp, pop_3, printer, vmnet, netbios_ns, urh_i, ssh, http_8001, iso_tsap, aol, sql_net, shell, supdup, auth, whois, discard, sunrpc, urp_i, Rje, ftp, daytime, domain_u, pm_dump, time, hostnames, name, ecr_i, bgp, telnet, domain, ftp_data, nnsp, courier, finger, uucp_path, X11, imap4, mtp, login, tftp_u, kshell, private, http_2784, echo, http, ldap, tim_i, netbios_dgm, uucp, eco_i, Remote_job, IRC, http_443, red_i, Z39_50, Pop_2, gopher, Csnet_ns
3	Flag (4)	OTH, S1, S2, RSTO, RSTRs, RSTOS0, SF, SH, REF, S0, S3

Afterwards, the rest of 38 features has numerical value with a large difference between maximum and minimum values. In particular, the feature "*duration*" where the maximum value is 42908 and the minimum is 0 in training set. Therefore, we performed a min-max normalization for mapping, these features are normalized to restrict the range of the values between 0 and 1 using Eq. 4.1:

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

where x = (x_1,..., x_n) is a number of input values and x_i is the respective value of the original feature. Besides, $max(x)$ and $min(x)$ represent the maximum

and the minimum values in x given its range. Consequently, the 41 features from the original dataset are transformed into 122-dimensional features.

4.3 Performance metrics

All the classifiers used in this research were evaluated using well-known methods to evaluate the classifiers performance. They are based on the confusion matrix of binary problems having positive and negative class values, which in this case normal and attack classes. The Table 4.5 shows the two class confusion matrix.

Table 4.5: Confusion matrix for binary classification problems

		Predicted	
		Attack	Normal
Actual	Attack	TP	FN
	Normal	FP	TN

Accuracy is used as a main evaluation indicator to measure the performance of the proposed IDS framework. As well as, we estimate several other performance metrics which is widely used to assess a models significance.

- **Accuracy**: the proportion of correct classification records to the total number of network records:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

- **Precision**: the proportion of correct classified records to the total number of records classified as an attack:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

- **Recall**: the proportion of correct classified records to the total number of actual attack records:

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

- **F1-Score**: the harmonic mean of precision and recall, which express the performance of the proposed approach:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.5}$$

where True Positives (TP) represent the actual attack records classified as a attack; True Negatives (TN) represent the actually normal records classified as an normal; False Positives (FP) are the actually normal records misclassified as a attack; False Negatives (FN) are the actually attack records misclassified as an normal.

In addition, Area under ROC Curve (AUC) is a useful metric even for datasets with highly unbalanced classes and measure of how well a binary classifier can perform predictions of labels. It represents the relationship between TPR and FPR.

4.4 Results and Discussion

During the sparse autoencoder feature extraction, we use the Gradient Descent Optimizer with a learning rate 10^{-3} to minimize error. Batch size of 128 and number of epochs is 100. In order to increase the detection result, we take experiments with changing the hidden layer dimension from 40 to 90. When the hidden nodes of 60 gives us the best results among of all evaluation metrics. After dimensional reduction step, our feature dimension mapped into 60 reduced features. Thus, the RNN classifier model has 60 input vectors and 2 output vector with learning rate 0.01. The experiments show that the RNN model achieved the 80.0% of accuracy when the training epochs are 100. Table 4.6 shows the confusion matrix of the RNN model on the testing set.

Table 4.6: Confusion matrix of the RNN model on testing data

		Predicted	
		Attack	Normal
Actual	Attack	8818	4023
	Normal	474	9229

We find that the proposed model has higher accuracy on the testing set when the hidden nodes are 40 in RNN classifier model. In that case, we use learning rate with 0.01, and the epoch number is 100. We observe the classification accuracy on the NSL-KDD dataset as shown in Figure 4.3. The

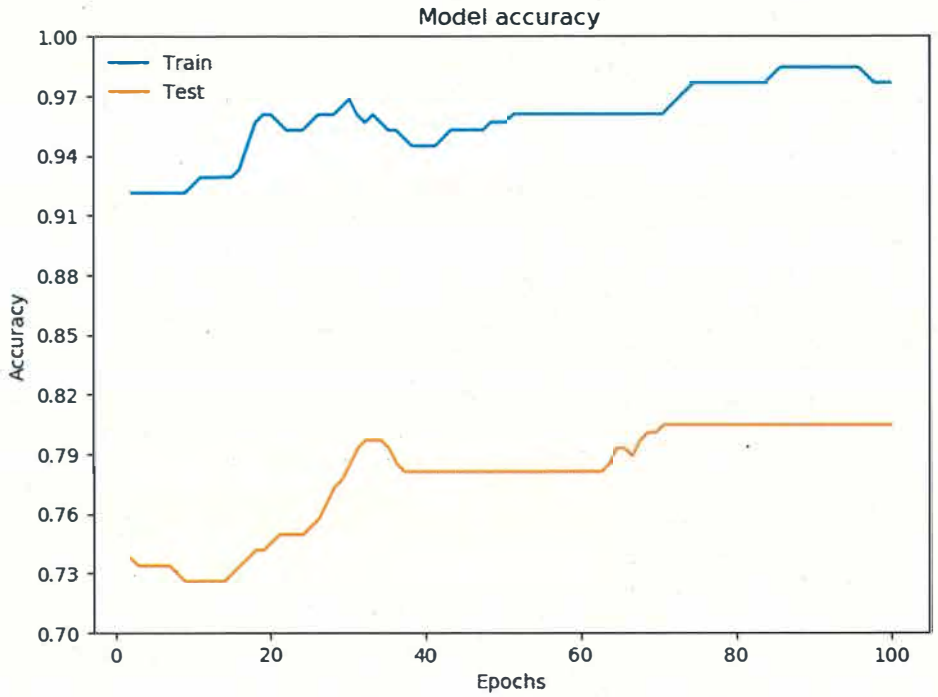

Figure 4.3: The accuracy of the proposed model

performance of our method is compared with the previous studies [15] [10] [27] [20] in classification accuracy on the NSL-KDD dataset as depicted in Table 4.7.

Table 4.7: Accuracy comparison with previous research methods

Methods	**Feature extractor**	**Classification**	**Accuracy (%)**
STL[15]	SAE	Softmax	88.39
DNN[10]	DNN	DNN	75.75
RNN[27]	RNN	RNN	83.29
CNN[20]	CNN	CNN	79.1
Our method	**SAE**	**RNN**	**80.0**

4.5 Conclusion

In this study, we have implemented IDS based on deep learning approach, which combines the sparse autoencoder and recurrent neural network. The first method to extract the low dimensional representation of the dataset while reducing a data dimension. In the following, we utilize the RNN algorithm to detect network attacks. In final, we assess the performance of the proposed method on the NSL-KDD dataset. The chapter results were published in [24].

Chapter 5

A two-stage NIDS framework

5.1 Overview

This chapter detailed the implementation of the network IDS framework. The framework consists of two stages: The first stage is a feature extraction stage which has two steps: unsupervised pre-training and supervised fine-tuning.

The first step is an unsupervised pre-training step that learns the typical patterns of the network traffic using a single-layer Sparse Autoencoder (SAE) algorithm without target label. Consequently, the second step is a supervised fine-tuning step that extracts the primary features the using preceding optimal parameters. Because of network traffic with high dimensionality, the computation time and training time certainly highly probably. By using dimension reduction techniques, it can remove redundant and duplicated features of input data. Then followed by extract appropriate features from the training and testing set using prior learned feature extractor.

The second stage is a an intrusion detection stage based on Long Short-Term Memory (LSTM) with softmax classifier to identify the traffic as normal or attack.

5.2 Proposed framework

In this section, we describe a details of our proposed framework.

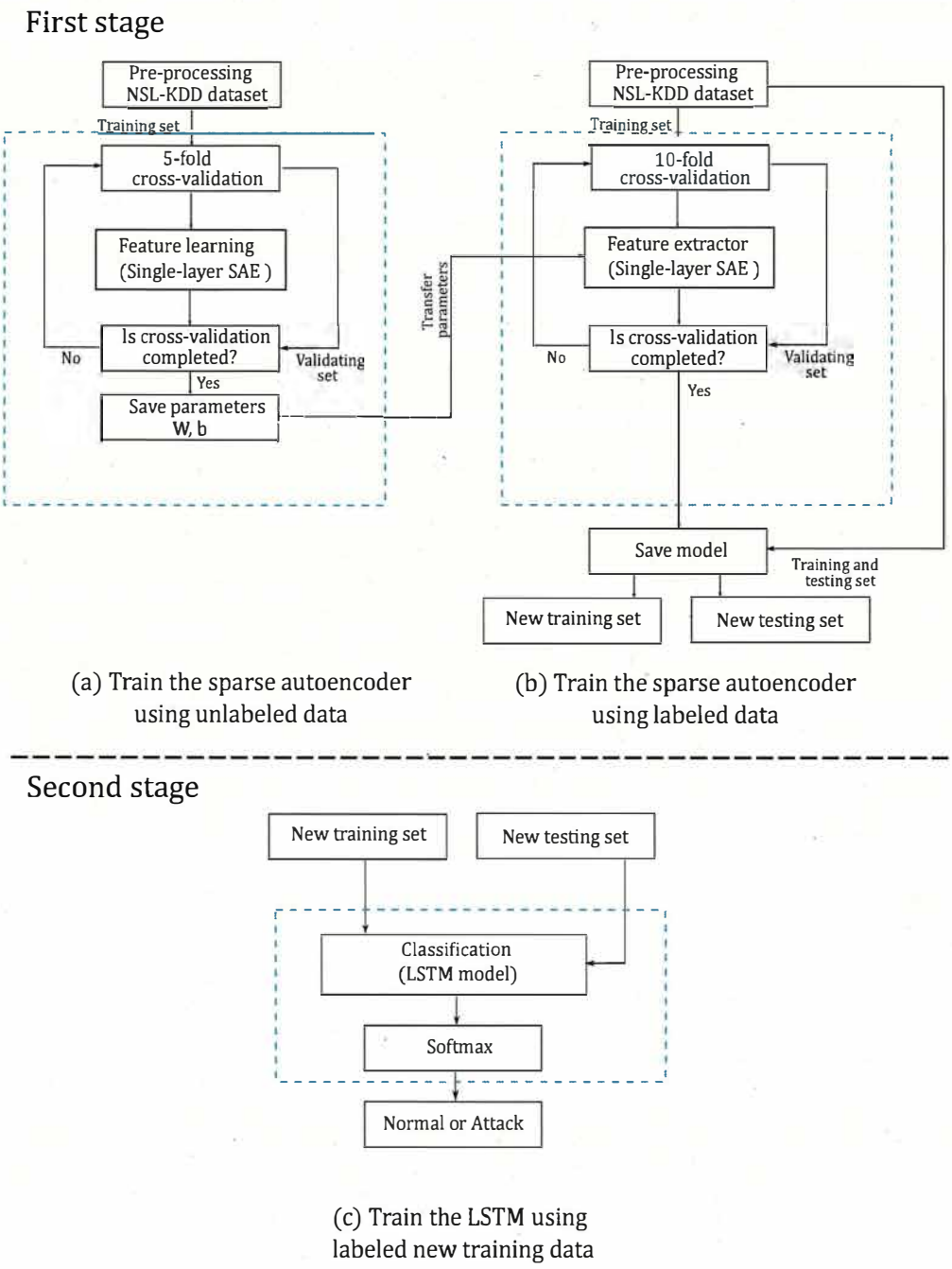

Figure 5.1: The proposed IDS framework based on SAE-LSTM for intrusion detection. (a) Unsupervised pre-training (b) Supervised fine-tuning (c) Classification

As illustrated in Figure. 5.1, the process of the SAE-LSTM framework are following steps:

Step 1. Unsupervised pre-training

The unsupervised learning algorithms can learn the typical pattern of the network and can report anomalies without any labeled dataset. Therefore we train a single-layer feature learner SAE on training set only in unsupervised manner using 5-fold cross-validation, it involves finding the optimal network parameters W, b by minimizing discrepancy between input data and its reconstruction data. After the network learned optimal values for W and b, save the network parameters.

Step 2. Supervised fine-tuning

The unsupervised pre-training can extract informative features that support intrusion detection. However, these features have not been identified with specific classes. Then we need to do is to further identify these features with supervised fine-tuning using labeled dataset. Once we find these parameters in previous step, we removed the decoder network and the encoder network is retained to produce primary features. As shown in Figure 3.3, the encoder network helps to get new training and testing data with low dimension from the pre-processed dataset. In other words, training and testing set fed into the feature extractor model which has preceding optimal parameters. We trained a feature extractor neural network until the minimum error is obtained, save the model. Consequently, using this model we can be obtain new training and testing set with primary features h_1, h_2, h_3 ... h_k which can well represent the input data.

Step 3. Classification

Train a LSTM model using the new training set to obtain a function that performs prediction of the intrusion detection. Then, apply new testing set into the prediction model, and the LSTM model can classifies the testing set as normal or attack. The details of each part are given below.

5.3 Experimental study

In this study, all the experiments were implemented using Python on a Tensorflow [45] 1.13.1 toolkit and tested on an Intel Core^{TM} i7 machine with GeForce GTX 1080 GPU.

5.3.1 Finding optimal hyperparameters in SAE

Our first experiments are conducted to study development of the feature extractor model named SAE. As a result of Sect. 4.2.1, a feature dimension of the training and testing set has been transformed into 122 dimension. Thus, the feature extractor SAE model has 122 input units and 80 hidden units to extract the good representation of input data.

Table 5.1: Hyperparameters for training SAE

Hyperparameter	Values	Selected value
λ	[1e-06, 2e-05, 1e-03, 1e-02]	1e-06
ρ	[0.02, 0.05, 0.1, 0.2, 0.3]	0.1
β	[3, 4, 5, 6]	3

We present our experimental results on the impact of hyperparameters on performance. First, we will evaluate the effects of hyperparameters (λ, ρ, β) in the L2 regularization and sparsity penalty term using cross-validation on the NSL-KDD training set. In this time, we utilized k-fold cross-validation ($k = 5$) to search the best value for these hyperparameters. As show in Table 5.1, our SAE model running 5-fold cross-validation method for combination of every value of hyperparameters. Figure 5.2 plots the cross validation error on different number of ρ, and the ρ in penalty term is set to be 0.1. Because our feature extractor model is very sensitive to this hyperparameter in a small range (ρ<0.02). We assess the performance of the fitted model on the validation set, the SAE model obtained the lowest cross validation error when $\lambda = 1e-6$, $\rho = 0.1$, $\beta = 3$ run over 50 epochs with learning rate equal to 0.01. Figure 5.3 and Figure 5.4 shows the cross validation error result on small range of λ and β. Once selecting proper hyperparameter values, we re-train

the SAE model using optimal hyperparameter values on training set with a label. Thus, we can extract the appropriate features from the input, and a data dimension is gradually reduced from 122 to 80.

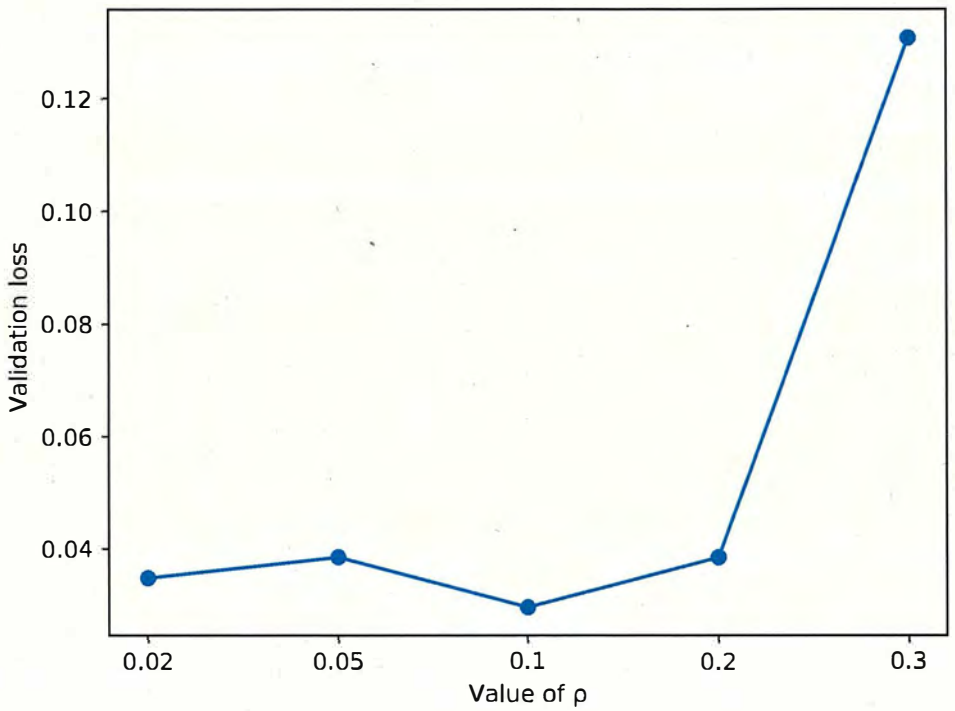

Figure 5.2: Validation loss on different value of hyperparameter ρ

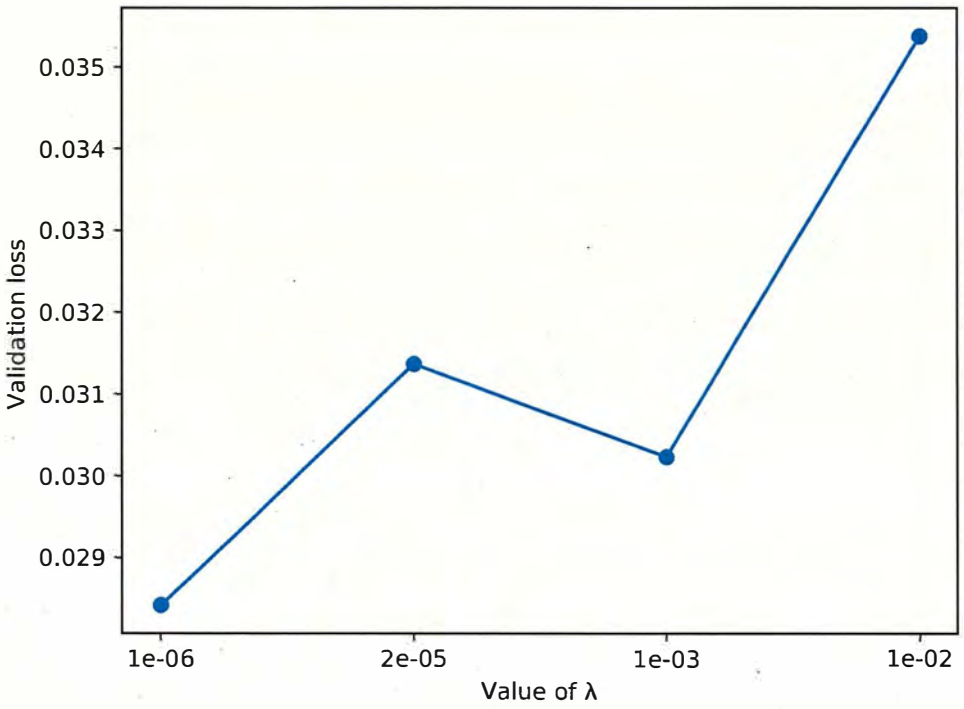

Figure 5.3: Validation loss on different value of hyperparameter λ

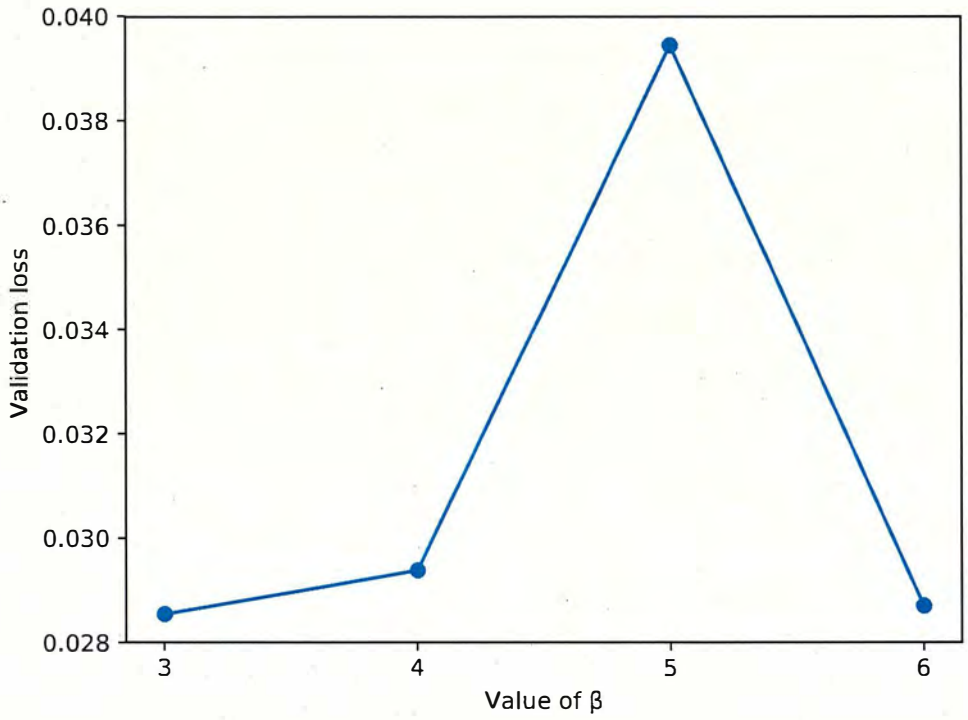

Figure 5.4: Validation loss on different value of hyperparameter β

5.3.2 Experiment on binary classification

To illustrate the performance of the feature extractor model, T-Distribution Stochastic Neighbor Embedding (t-SNE) [46] was used to visualize the feature vectors of SAE model. The visualization technique t-SNE can map high-dimensional feature vectors into 2 dimensions and show distributions of the high-dimensional feature vectors. After unsupervised pre-training the SAE model with a large number of unlabeled data, we can feed it with labeled data to extract feature vectors from the hidden layer. Then we obtained feature vectors after using labeled data to do feed-forward inference, and we used to visualize these feature vectors using the t-SNE. In order to compare, we prepared visualization set which consists of 500 samples are randomly selected from each classes (normal, attack). Figure 5.5 is visualization result from the

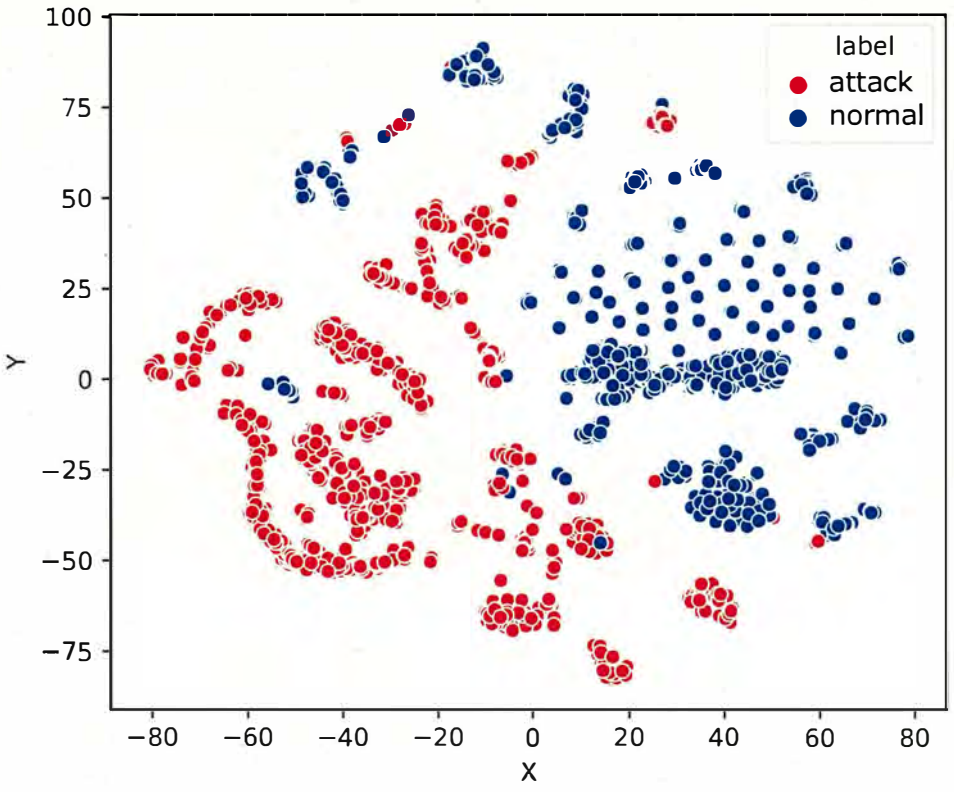

Figure 5.5: Visualization result of raw features 122 for binary classification

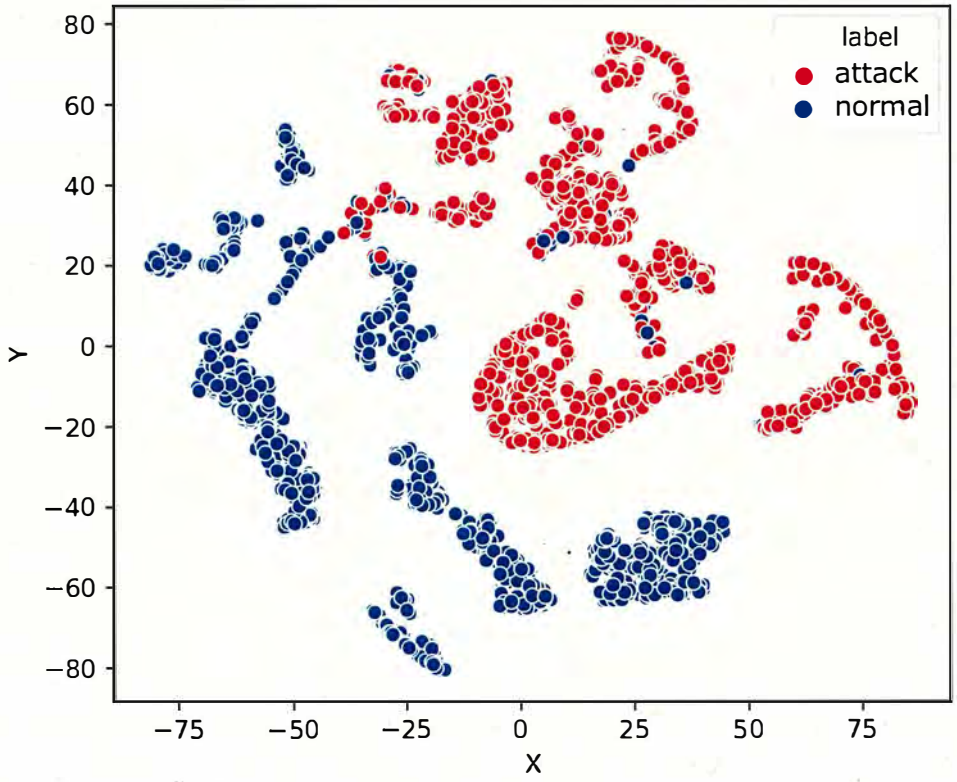

Figure 5.6: Visualization result of reduced features 80 for binary classification

raw features, while Figure 5.6 is visualization result from the feature vectors of the feature extractor model. For each class, we can observe clearly separated clusters, Figure 5.5 shows that two different classes of raw features mainly cluster into two parts. In Figure 5.6, the distributions of feature vectors of SAE model is similar to the distributions of raw features. For this, we can preliminary think that the SAE model under unsupervised pre-training and supervised fine-tuning can learns potential features from the network traffic.

In the following, we evaluate the effectiveness of the LSTM model which has 80 input units and 2 output units. We used to apply the 10-fold cross-validation method to validate results in our LSTM model to prevent over-fitting. To demonstrate the effectiveness of the SAE-LSTM framework, we attach the softmax classification to the output layer of the LSTM model. Figure 5.7 shows the effect on the performance of the LSTM model taken a

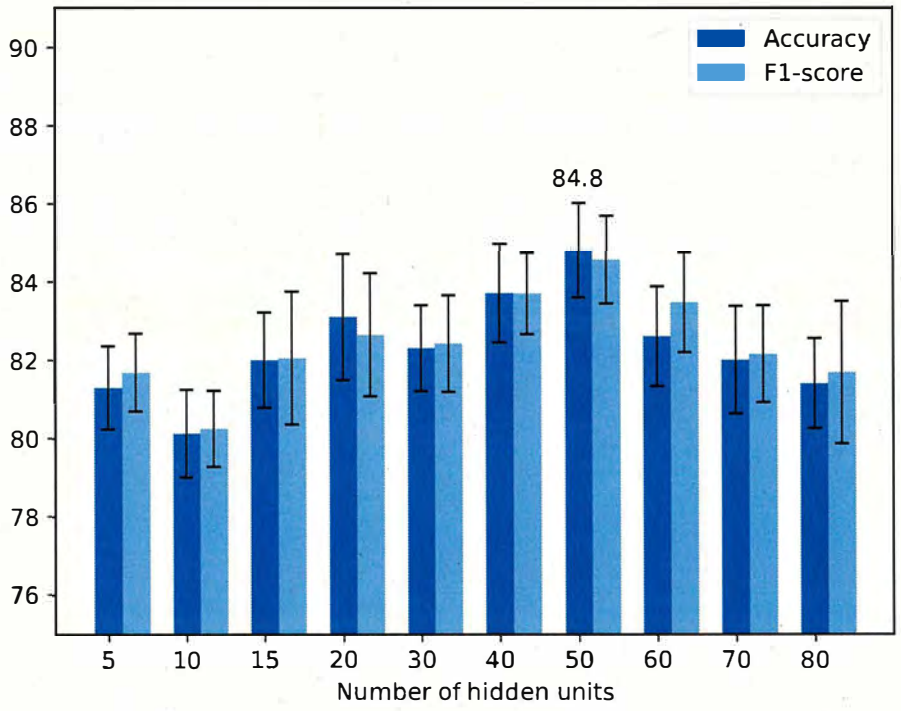

Figure 5.7: Effect of different hidden units for the SAE-LSTM on the testing dataset

different number of hidden units on new testing data, including the mean accuracy and standard deviation for each point estimated over 10 trials. From the result, the proposed framework achieves excellent performance on hidden units of 50, when running over 40 epochs with learning rate 0.01 As presented in Figure 5.7, our classification accuracy is 84.8%±1.21% on the testing set. As well as, our proposed framework achieved 84.5%±1.19% of f1-score. It give us the best classification results for intrusion detection. Moreover, the training error of the LSTM model as depicted in Figure 5.8, and it considered highly probable without overfitting. In the Figure 5.9, Receiver Operating Characteristic (ROC) curves are presented, with respect to true positive rate and false positive rate. The area under the ROC Curve (AUC) is computed at 86.2%.

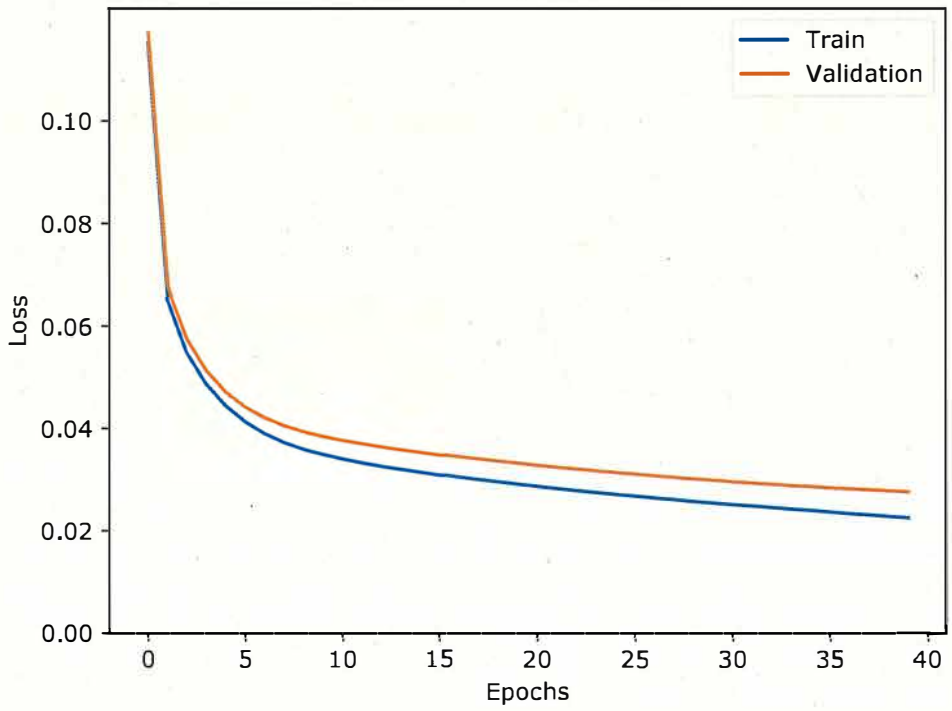

Figure 5.8: Training error of the LSTM model on hidden units 50

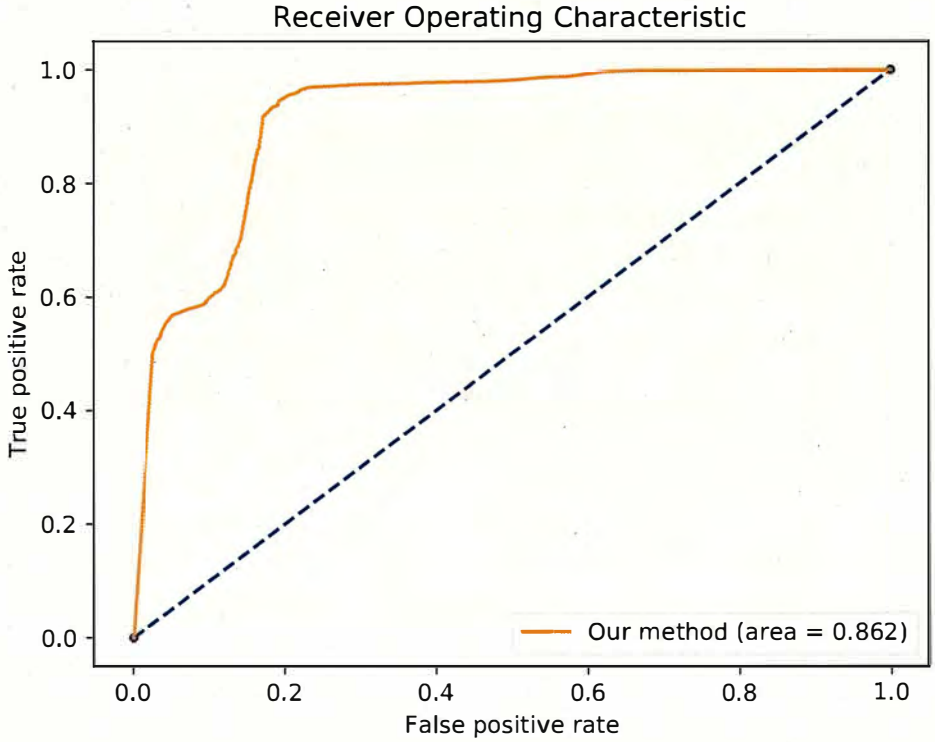

Figure 5.9: ROC curve of the LSTM model on hidden units 50

5.3.3 Experiment on 5-class classification

As well as, we also visualized the feature vectors learned by the SAE model in 5 classes as shown in Table 4.3. In the first, we sampled 200 samples are randomly selected from each attack class (DoS, Probe, R2L, and U2R) and normal data from the raw featured dataset. In totally, 1000 samples are visualized in Figure 5.10. In the next, we built SAE model to extract appropriate features for 5-class classification. Same as above mentioned, unsupervised pre-training the SAE model with a large number of unlabeled data, and transfer the learned parameters to the supervised fine-tuning SAE model. Then we retrain the model with labeled data to extract feature vectors from the feature layer. Finally we received the feature vectors for 5-class dataset, and we used to visualize these feature vectors using the t-SNE. In Figure 5.11 illustrates the visualization result of representation vectors of the feature extractor.

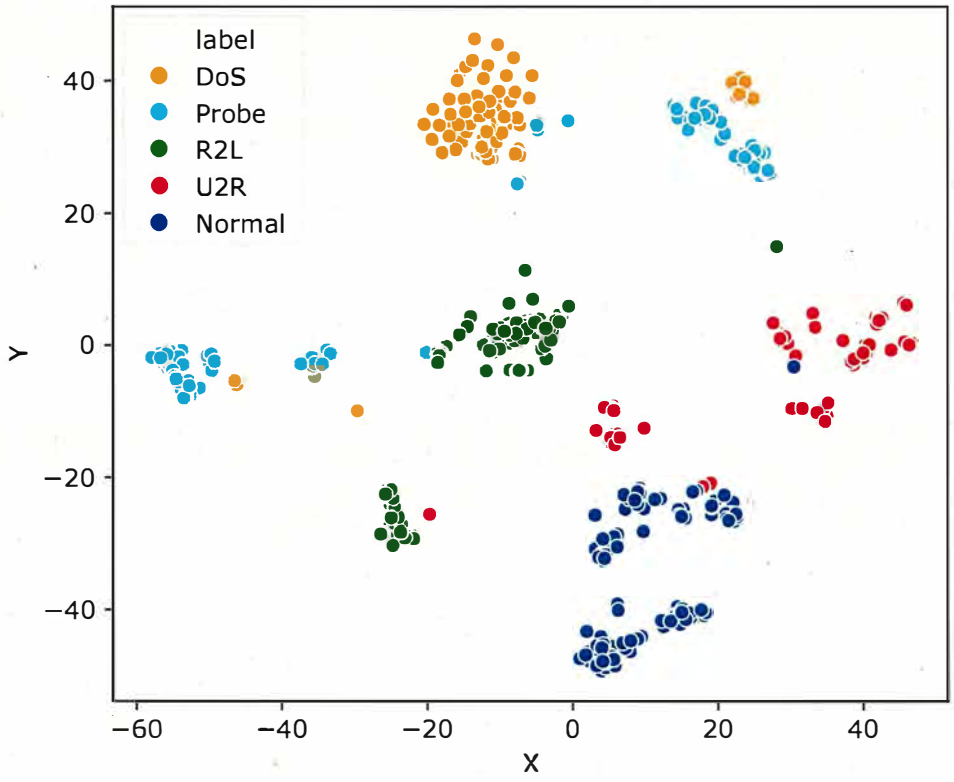

Figure 5.10: Visualization result of raw features 122 for 5-class classification

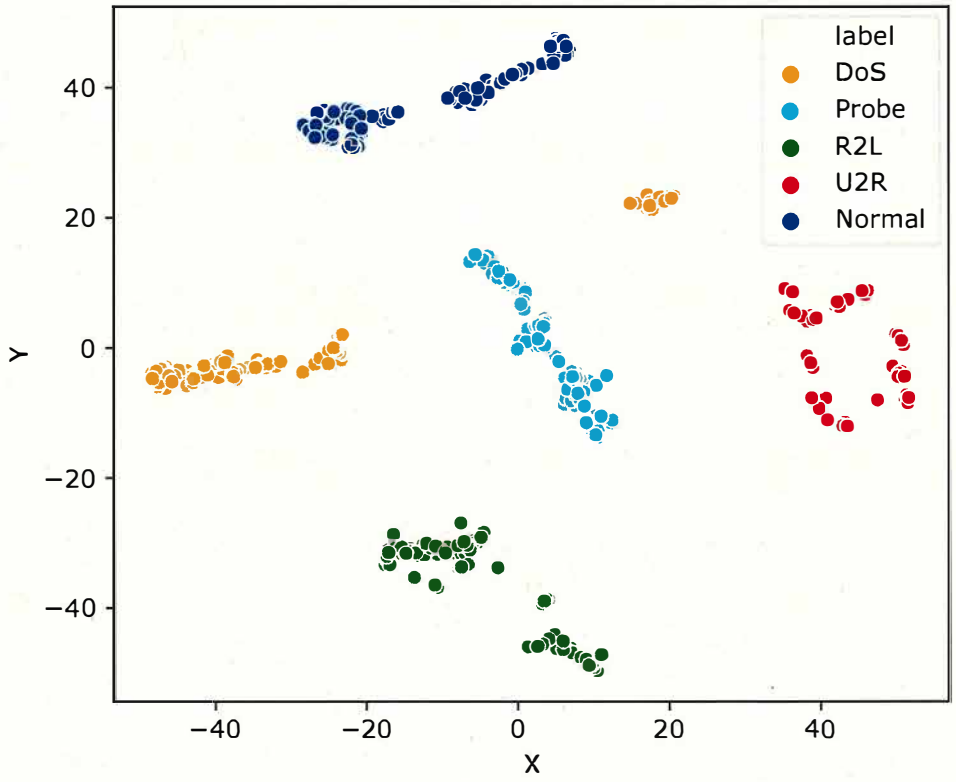

Figure 5.11: Visualization result of reduced features 80 for 5-class classification

In general, the visualization results demonstrate that our feature extractor model are quite good to learn primary features from the network traffic.

In the following, we evaluate our proposed IDS framework for 5-class classification which includes normal and four type of attacks named DoS, Probe, R2L, and U2R. It outlined in previous section. According to the illustrated in Figure 5.1, our feature extractor SAE model extracts the potential feature vectors from the raw feature dimension, however output dimension of supervised fine-tuning model has 5-dimensional vectors. Afterward, train the LSTM model using new training set, and we estimated the confusion matrix and ROC curve with AUC for each 5 classes.

The results obtained from the 5-class analysis of the intrusion dataset by our proposed IDS framework. We draw confusion matrix to further evaluate the intrusion detection of the proposed IDS framework, which are depicted in

Figure 5.12. From the result, three classes of traffic (DoS, Probe and Normal) are classified well. However, the effects of the proposed IDS framework for the R2L and U2R classes are not as good as than other three classes. Table 5.2 shows the detection rate of the different attack types.

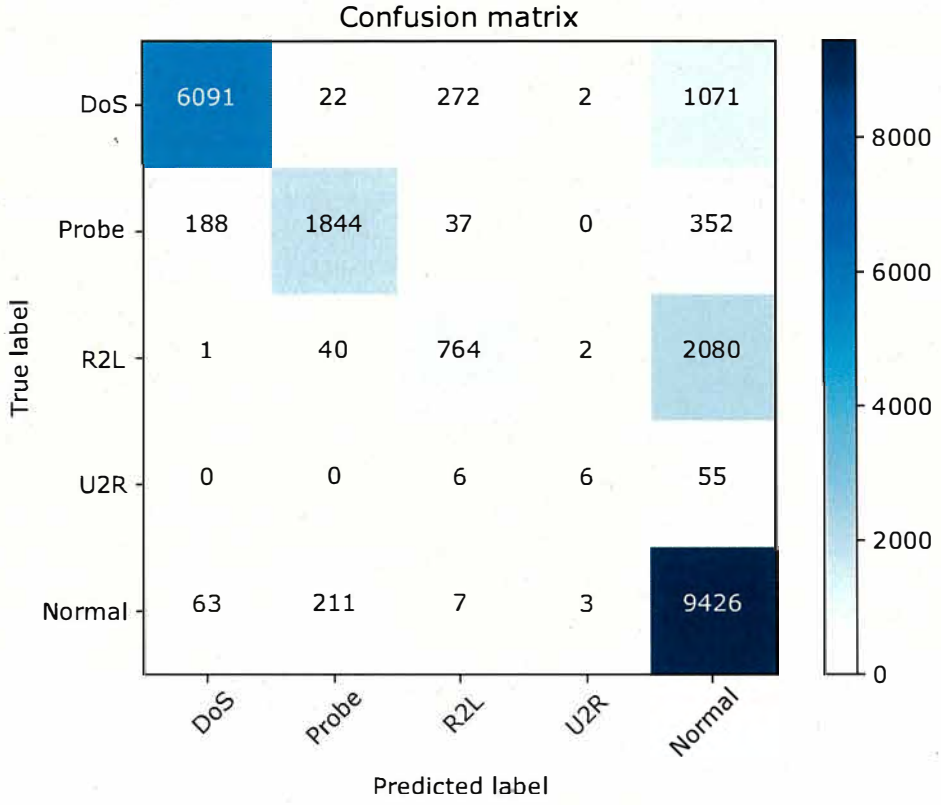

Figure 5.12: Confusion matrix for the 5-class classification of SAE-LSTM

Table 5.2: Result of the detection rate for the 5-class

Traffic type	**Detection rate (%)**
DoS	81.6
Probe	76.1
R2L	29.5
U2R	8.9
Normal	97

We represented ROC curves with AUC values to evaluate the proposed IDS framework in 5-class.

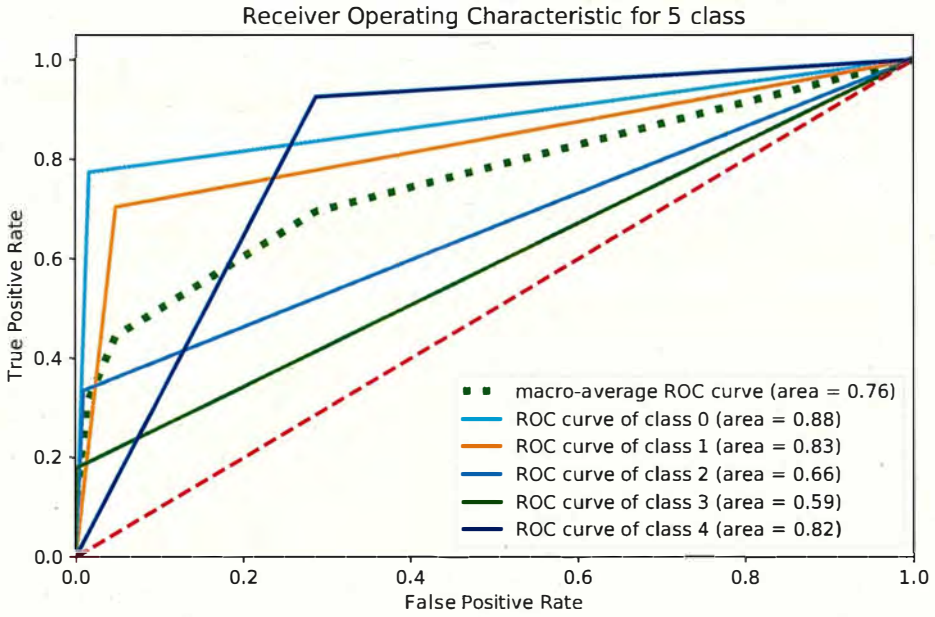

Figure 5.13: ROC curve for the 5-class classification of SAE-LSTM

5.4 Results and Discussion

In this section, we compare the results of our proposed framework with the results of the literature used by different researchers. Table 5.3 reports the comparison average accuracy and f1-score of detecting attack between the proposed framework and in advance IDS models on the NSL-KDD dataset. In terms of accuracy, our model outperforms all listed methods and the proposed framework consistently achieves comparable results with state-of-the-art methods.

Tavallaee *et al.* [42] claimed that their NSL-KDD dataset performed on several machine learning algorithms. The results shown in Table 5.3 are inferior to those of our proposed method. Having irrelevant features in the dataset can decrease the performance of a model. Therefore, a couple of research works introduce the feature selection by removing irrelevant features.

Table 5.3: Performance comparison with other published methods for NSL-KDD dataset

References	Methods	Accuracy	F1-score
Tavallaee [42]	J48	81.05%	-
	Naive Bayes	76.56%	-
	NB Tree	82.02%	-
	Random Forest	80.67%	-
	Multilayer Perceptron	77.41%	-
	SVM	69.52%	-
Ingre [10]	Artificial Neural Network	81.2%	-
Tang [14]	Deep Neural Network	75.75%	75%
Pajouh [11]	LDA + Naïve Bayes	82%	-
Yin [27]	Recurrent Neural Network	83.28%	-
Li [20]	Convolutional Neural Network		
	ResNet50	79.14%	79.12%
	GoogLeNet	77.04%	76.50%
Proposed framework	**SAE-LSTM**	**84.8±1.21%**	**84.5±1.19%**
Javaid [1] *[15]*	*Self-Taught Learning*	*88.39%*	*90.4%*

[1] The method only evaluated on training data for both training and testing set

[2] The '-' indicates that there is no experiment on corresponding metrics.

Ingre *et al.* [10] are proposed ANN model that reduced the feature set by removing almost all zero values from the dataset. The paper achieved an accuracy of 81.2% on the testing set. As a similar, a method described by Tang *et al.* [14] only attempts to use six features (*duration, protocol type, src bytes, dst bytes, count and crv count*) from 41 features in the SDN environment. From their experiments, the result not good enough to detect from some attacks. For example, *num_failed_logins* feature stops password guessing attacks by

locking the account after a set number of failed login attempts. However, the paper by presented Pajouh *et al.* [11] proposed the two-tier network anomaly detection approach using Linear Discriminant Analysis (LDA) for dimension reduction. The paper was obtained 82% of detection rate. Therefore, we tried to discover the effective features of the dataset through the non-linear mapping while reducing the data dimension.

Several deep learning methods are proposed for the intrusion detection without using feature extraction stage. Yin *et al.* [27] implemented RNN-IDS using recurrent neural network. The authors obtained the highest classification accuracy at 83.28% on the testing data. A paper introduced by Li *et al.* [20] uses CNN that adopts novel representation learning methods of graphic conversion for intrusion detection. The method of transforming the standard NSL-KDD dataset data form into 8*8 gray-scale images is introduced. They used the ReSNet50 and GoogLeNet network as CNN models. The performance of this study are obtained 79.14% of accuracy and 79.12% of f1-score on testing data.

Javaid *et al.* [15] further introduced more advanced feature extraction method based on Self-Taught Learning (STL). The STL has proven that an effective feature extraction of intrusion data, and they achieved higher accuracy of 88.39%. However, they present the evaluation results on the training data for both training and testing using 10-fold cross-validation.

In summary, these comparisons show that our proposed framework performs better than existing studies expected STL [15] in the classification accuracy which proves the effectiveness of our framework. Furthermore, experimental results validated that our feature extraction SAE model significantly effected to improve the performance of this work.

5.5 Limitation

For specific class classification, in particular, the detection rate of the 5-class classification represents as not good as in R2L and U2R classes. This outcome occurs because of there are too few training instances in the training set as can seen in Table 4.1.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude our thesis with a summary of accomplishments and provide insights for the future extensions of some of these works.

6.1 Conclusion

This thesis was aimed to develop an effective framework combining single-layer Sparse Autoencoder (SAE) and Long Short-Term Memory (LSTM) for network intrusion detection system. Initially, the feature extractor model proposed to extract the most relevant features for use in representing the data. The proposed model determines an approximation to the identity function, so as to output data that is similar to their input data. In other words, the function involves finding the optimal network parameters weight, biases by minimizing the discrepancy between input and its reconstruction data. Furthermore, we consider to optimize hyperparameter values of the feature extractor model using the 5-fold cross-validation method on training set, it can help to identify the good representations from the raw input data. In the following, the LSTM method proposed for classifying network traffic as normal or attack. Then finally, we evaluate the effectiveness of the proposed IDS framework on the benchmark NSL-KDD dataset. The experimental result shows that the two-stage IDS framework achieved a higher accuracy rate it outperformed other similar studies. As well as, our SAE model can extract effective features leads to obtain a good performance and improve the LSTM

model classification accuracy. In conclusion, our proposed IDS frameworks works well in binary classification, however 5-class classification was not good enough.

6.2 Future Work

In the future study, we will emphasize on to increase the model prediction performance of the current models and evaluate their performance on more attack types. Moreover, we need to more focused on the detection performance of the imbalanced dataset.

Bibliography

[1] C. A. Statistics, "Hackmageddon," *Avaialble online: http://hackmageddon. com/category/security/cyberattacks-statistics*, 2015.

[2] D. Sumeet and D. Xian:, *"Data mining and machine learning in cybersecurity"*. Auerbach Publications, 2016.

[3] V. G. Fixed, "Mobile internet traffic forecasts. cisco," 2017.

[4] I. Symantec, "Internet security threat report appendices," 2019.

[5] M. Roesch *et al.*, "Snort, network intrusion detection/prevention system," 2011.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–72, 2009.

[7] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *2018 10th International Conference on Cyber Conflict (CyCon)*, pp. 371–390, May 2018.

[8] W. Zong, Y.-W. Chow, and W. Susilo, "Dimensionality reduction and visualization of network intrusion detection data," in *Information Security and Privacy* (J. Jang-Jaccard and F. Guo, eds.), (Cham), pp. 441–455, Springer International Publishing, 2019.

[9] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, and F. Jasmir, "Automatic features extraction using autoencoder in intrusion detection system," in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pp. 219–224, Oct 2018.

[10] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *Signal Processing And Communication Engineering Systems (SPACES)*, pp. 92–96, IEEE, 2015.

[11] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, pp. 61–74, 2015.

[12] Y. Hamid, V. R. Balasaraswathi, L. Journaux, and M. Sugumaran, "Benchmark datasets for network intrusion detection: A review," *International Journal of Network Security*, vol. 20, no. 4, pp. 645–654, 2018.

[13] N. R. Sabar, X. Yi, and A. Song, "A bi-objective hyper-heuristic support vector machines for big data cyber-security," *IEEE Access*, vol. 6, 2018.

[14] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*, pp. 258–263, IEEE, 2016.

[15] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, 2016.

[16] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015.

[17] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[18] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short-term memory recurrent neural network classifier for intrusion detection," in *Platform Technology and Service (PlatCon), 2016 International Conference on*, pp. 191–195, IEEE, 2016.

[19] T.-T.-H. Le, Y. Kim, H. Kim, *et al.*, "Network intrusion detection based on novel feature selection model and various recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, p. 1392, 2019.

[20] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *International Conference on Neural Information Processing*, pp. 858–866, Springer, 2017.

[21] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[22] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223, 2011.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[24] Z. Kherlenchimeg and N. Nakaya, "Network Intrusion Classifier Using Autoencoder with Recurrent Neural Network," in *The Fourth International Conference on Electronics and Software Science (ICESS2018)*, pp. 94–100, 2018.

[25] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3854–3861, May 2017.

[26] 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, 2015.

[27] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[28] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," tech. rep., Technical report, 2000.

[29] F. Garzia, M. Lombardi, and S. Ramalingam, "An integrated internet of everything—genetic algorithms controller—artificial neural networks framework for security/safety systems management and support," in *2017 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, IEEE, 2017.

[30] D. W. Vilela, A. D. P. Lotufo, and C. R. Santos, "Fuzzy artmap neural network ids evaluation applied for real ieee 802.11 w data base," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, 2018.

[31] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 807–819, 2013.

[32] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[33] J. Heaton, "Artificial intelligence for humans, volume 3: Deep learning and neural networks; heaton research," *Inc.: St. Louis, MO, USA*, 2015.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[35] A. Ng, "Sparse autoencoder, vol. 72," *CS294A Lecture Notes*, 2011.

[36] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[37] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[38] 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*, pp. 6645–6649, May 2013.

[39] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

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

[41] I. N.-K. D. Set, "University of new brunswick est. 1785."

[42] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.

[43] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *arXiv preprint arXiv:1806.03517*, 2018.

[44] S. Selvakumar and S. Bhattacharya, "Multi-Measure Multi-Weight Ranking Approach for the Identification of the Network Features for the Detection of DoS and Probe Attacks," *The Computer Journal*, vol. 59, no. 6, pp. 923–943, 2016.

[45] M. Abadi, A. Agarwal, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[46] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

List of Publications

- Zolzaya Kherlenchimeg and Naoshi Nakaya, "A Deep Learning Approach Based on Sparse Autoencoder with Long Short-Term Memory for Network Intrusion Detection," IEEJ Transaction on Electronics, Information and System (C)", vol. 140, no.6, 2020.
- Zolzaya Kherlenchimeg and Naoshi Nakaya, "Network Intrusion Classifier Using Autoencoder with Recurrent Neural Network," In *The Fourth International Conference on Electronics and Software Science (ICESS2018)*, pp. 94-100, 2018.