Comparative Analysis of Deep Learning for Network Intrusion Detection Systems

---

Brenton Budler
1827655

Supervisor:
Dr. Ritesh Ajoodha

A research proposal submitted in partial fulfillment of the requirements for the degree of Bachelor of Science Honours (Computer Science)

in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

2 August 2021

Declaration

I, Brenton Budler (Student Number 1827655), declare that this proposal is my own, unaided work. It is being submitted for the degree of Bachelor of Science Honours (Computer Science) at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.

Brenton Budler
1827655
2 August 2021

Abstract

Detecting network intrusions is an imperative part of the modern cybersecurity landscape. Over the years, researchers have leveraged the ability of Machine Learning (ML) to identify and prevent these attacks. Recently there has been an increased interest in the applicability of Deep Learning (DL) in the network intrusion detection domain. However, these DL-based Network Intrusion Detection Systems are being proposed and evaluated using outdated datasets which are not representative of real-world network traffic. By performing a comparative analysis of five popular DL approaches on both a legacy and a modern dataset, this research aims to establish a new baseline for future research in the application of DL for network intrusion detection and provide deeper insights into the performance of these models in detecting and classifying network intrusions.

Acknowledgements

I would like to thank the University of the Witwatersrand for their continuous involvement in my academic and personal development as well as Dr. Ritesh Ajoodha for his wisdom, guidance and unwavering support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the years, the use of the internet and other interconnected networks has grown exponentially. This rapid development of online technologies has, however, been accompanied by an equally rapid increase in cyberattacks. Globally, Cybersecurity Ventures predicts that cybercrimes will cost companies $10.5 trillion by 2025. In South Africa, research by Accenture indicates that R2.2 billion is already lost to cyberattacks every year. These devastating consequences are a result of cybersecuirty breachs taking an average of 280 days to identify and contain as indicated in IBM's *Cost of Data Breach Report 2020*.

The only solution to these constantly evolving attack scenarios is cybersecurity systems built on the foundations of Artificial Intelligence (AI) which can leverage the ability of Machine Learning (ML) to identify anomalies, detect and prevent these attacks. Once such system is a Network Intrusion Detection System (NIDS) which monitors incoming and outgoing network traffic in an attempt to identify threats [Yan and Han 2018].

NIDSs are divided into two major categories; signature-based and anomaly-based [Javaid *et al.* 2016]. Signature-based systems, also known as misuse-based systems, operate by comparing network traffic with a stored database of known attack patterns and flagging potential threats based on their similarity to these attacks [Wang *et al.* 2017]. The most prevalent drawback of signature-based systems is the inability to identify zero-day (novel) attacks [Yan and Han 2018]. The comparison they perform to identify attacks is also computationally expensive and relies on storing and maintaining a database of all known attack patterns [Dong *et al.* 2019].

On the other hand, anomaly-based systems first determine a benign network traffic baseline and then identify any network activity which deviates significantly from this baseline as a potential threat [Marir *et al.* 2018]. The ability of anomaly-based systems to identify unknown attacks has led it to become the most widely used approach in identifying network intrusions [Alrawashdeh and Purdy 2016].

ML, a sub-field of AI , involves the construction of systems that improve automatically through experience [Jordan and Mitchell 2015]. In recent years, many researchers have shown the ability of ML to effectively solve the anomaly detection and intrusion classification problem [Yin *et al.* 2017]. Unfortunately, the construction of these traditional ML models relies heavily on feature engineering [Elsayed *et al.* 2020]. This feature engineering is both time-consuming and requires a high-level of expert interaction [Shone *et al.* 2018].

An extension of ML known as Deep Learning (DL), involves the creation of computational models comprised of multiple processing layers [LeCun *et al.* 2015]. These multi-layer models do not rely as heavily on feature engineering as they are able to automatically learn complex features and extract better representations of the data [Yin *et al.* 2017]. In the field of network intrusion detection, these abstract data representations allow models to identify anomalies faster and more effectively with little to no human interaction [Shone *et al.* 2018].

The focus of this research is an evaluation of different Deep Learning-based Network Intrusion Detection Systems on a modern dataset; the CSE-CIC-IDS2018 and a comparative analysis of these models with one another, as well as with their own performance on a legacy dataset, the NSL-KDD. A fair evaluation of these algorithms will provide deeper insights into which of them have withstood the test of time, and will provide a baseline for future research in the application of DL for network intrusion detection.

1.1 Related Work

Although research into the application of DL methods for intrusion detection is still in its early stages [Ahmad *et al.* 2021], several researchers have already begun experimentally evaluating DL algorithms in the network intrusion detection domain. The table below outlines some of these methods and the performance metrics achieved. The predominant evaluation metrics used in DL-based NIDS research are binary and five-class classification, represented in the table by ACC-2 and ACC-5 respectively. Area Under the Curve (AUC) and Detection Rate (DR) are also occasionally used to evaluate the performance of a model. Evaluation metrics are explored further in Section 2.3 - Analysis.

TABLE 1.1: Key contributions in the literature

Authors	**Dataset**	**Model**	**Metric**	**Value**
Alrawashdeh and Purdy [2016]	KDD Cup '99	DBN	ACC-5	97.90%
Javaid *et al.* [2016]	NSL-KDD	Sparse AE with Soft-max Regression	ACC-5	79.10%
Yin *et al.* [2017]	NSL-KDD	Recurrent NN	ACC-5	81.29%
Wang *et al.* [2017]	DARPA98	Deep Convolution NN and LSTM	ACC-2	99.69%
Shone *et al.* [2018]	NSL-KDD	Non-symmetric Deep AE with RF	ACC-5	85.42%
Marir *et al.* [2018]	NSL-KDD	DBN and Ensemble SVM	AUC	0.986
Yan and Han [2018]	NSL-KDD	Stacked Sparse AE	DR	89.84%
Al-Qatf *et al.* [2018]	NSL-KDD	Sparse AE and SVM	ACC-5	80.48%
Xiao *et al.* [2019]	KDD Cup '99	AE and Convolutional NN	ACC-5	94%

1.1.1 Data

The two most used datasets in NIDS research are the KDD Cup '99 and the NSL-KDD datasets [Yan and Han 2018]. The KDD Cup '99 dataset was generated in 1999 from the DARPA98 network traffic which was collected over 9 weeks in raw tdpdump format [Ahmad *et al.* 2021]. The KDD Cup '99 has been used as a benchmark in NIDS research for many years but suffers from the major drawback that about 75% of the records are redundant, leading to learning algorithms being biased towards attack records which appear more frequently [Javaid *et al.* 2016].

Due to the sheer volume of the KDD Cup '99 dataset (5 million training records and 2 million testing records), only 10% of the original data is conventionally used in evaluations to allow for reduced computation [Alrawashdeh and Purdy 2016] .

The NSL-KDD dataset was derived from the original KDD Cup '99 dataset and effectively solves the inherent redundant record problem [Yin *et al.* 2017]. It also partitions the records into different difficulty categories based on the number of traditional ML algorithms that are able to correctly classify the records.

Both datasets consist of 41 network traffic features which contain both host-based and time-based information. Each of the instances within the dataset is classified as either benign or as one of the four attack types; User to Remote, Denial of Service , Probe or Remote to Local [Ahmad *et al.* 2021].

Data pre-processing of both these datasets typically includes the scaling of numerical features using minimum-maximum normalization [Xiao *et al.* 2019]. Logarithmic scaling is also used by some researchers for features with high standard deviation [Hsu and Matsuoka 2020]. Binary and one-hot encoding are used to transform the categorical features into numerical features [Alrawashdeh and Purdy 2016].

The primary disadvantage of both the KDD Cup '99 dataset and the NSL-KDD dataset is that they are outdated. Network architectures have changed dramatically over the past 20 years and these older datasets are no longer representative of modern attack styles [Marir *et al.* 2018].

1.1.2 Models

In the literature, model construction is performed using an array of different tools including several Python frameworks such as TensorFlow [Yan and Han 2018], Weka [Yin *et al.* 2017] and Keras [Suwannalai and Polprasert 2020] as well as Matlab [Al-Qatf *et al.* 2018] and C++ [Alrawashdeh and Purdy 2016]. Model parameters are usually determined experimentally with researchers testing various network architectures to determine which parameters allow for the best performance without exorbitant training times [Shone *et al.* 2018].

The hybrid idea of using DL for feature extraction and ML for classification has become a popular trend as it reduces model complexity whilst ensuring accurate results [Ahmad *et al.* 2021]. Alrawashdeh and Purdy [2016] performed the DL feature extraction using a Deep Belief Network (DBN) while Javaid *et al.* [2016] , Yan and Han [2018] and Shone *et al.* [2018] used variants of an AutoEncoder (AE) to reduce feature dimensionality.

The classification task is predominantly performed using a multi-class Soft-Max Regression (SMR) [Javaid *et al.* 2016] although Support Vector Machines [Yan and Han 2018] and Random Forests [Shone *et al.* 2018] have also showed promising results.

Xiao *et al.* took a slightly different approach and performed dimensionality reduction using an AE, the data was then transformed into a two-dimensional matrix which was fed to a Convolutional Neural Network (CNN). The CNN is used to learn the optimal features which are then passed to a SMR to perform the classification.

Unlike the aforementioned models which use a DL algorithm for pre-training only, Yin *et al.* proposed a model which uses a Recurrent Neural Network (RNN) to extract better representation from the data and perform the classification. They compare their results with the classification ability of several algorithms and demonstrate that their model is able to increase accuracy and lower the False Alarm Rate (FAR).

Looking at the work that has been done to develop NIDS using DL, it is evident that although these algorithms have provided substantial increases in accuracy and detection rate, evaluating them on outdated datasets results in questionable conclusions being drawn. There is a crucial need for a fair evaluation of these models on a modern dataset, hereby analyzing their true performance in the modern cybersecurity landscape.

1.2 Problem Statement

In an ideal world a NIDS built using DL techniques could be evaluated on real-world network traffic data. However, due to company privacy issues this data is not publicly available [Marir *et al.* 2018]. The benchmark datasets which are employed when evaluating novel models, namely the KDD Cup '99 and NSL-KDD, are severely outdated. Modern researchers still validate the use of these datasets with the fact that it allows them to draw comparisons with other academic literature [Shone *et al.* 2018]. If future researchers continue to evaluate novel approaches using these outdated datasets, their conclusions will only become more disputable.

1.3 Research Aim

In an attempt to establish a new baseline for future researchers to compare their work, this research involves a comparative analysis of five popular DL-based NIDS on both a legacy benchmarket dataset, the NSL-KDD and a modern dataset, the CSE-CIC-IDS2018. This includes the construction, training, testing and evaluation of a Deep Neural Network, a Deep Belief Network, a Self-Taught Learning Model, A Convolutional Neural Network and a Long Short-Term Memory Recurrent Neural Network.

1.4 Research Question

Are Network Intrusion Detection Systems developed using Deep Learning techniques able to attain the same accuracy, detection rate and false alarm rate on modern network traffic data as they could on a legacy dataset?

1.5 Research Objectives

- **Data Preparation**: Ingest and preprocess both the NSL-KDD and CSE-CIC-IDS2018 datasets using the standard approaches found in the literature.
- **Model Construction**: Construct the five models that will be used in the comparative analysis.
- **Model Testing and Validation:** Test the models using the NSL-KDD dataset to ensure they are attaining results comparable to the ones attained in other research that utilize similar models.
- **Model Evaluation and Comparison:** Evaluate each models performance on the CSE-CIC-IDS2018 dataset using a range of evaluation metrics.
- **Result Documentation and Presentation:** Document the findings of the comparitive analyis in a research report and prepare a presentation of the work.

1.6 Overview

The rest of this document is structured as follows: Chapter 2 will delve further into the methodology that will be used to carry out this research. Section 2.1 includes a detailed outline of the data to be used in the comparative analysis. Section 2.2 gives a brief description of each of the models that will be constructed and evaluated. Section 2.3 outlines the various evaluation metrics to be used and Section 2.4 explores the hardware and software that will be utilized to construct, train, test and evaluate the different models.

Chapter 3 outlines the research plan and includes a week-by-week schedule of work and a description of potential difficulties. Lastly, Chapter 4 concludes with a brief summary of the proposal and final remarks.

Chapter 2

Research Methodology

In the previous chapter, academic literature related to Deep Learning-based Network Intrusion Detection Systems was explored. The problem that this research is attempting to solve was also stated along with the primary aim of this research and the objectives that need to be met in order to achieve this aim. This chapter will include a breakdown of each aspect of the proposed research including the data to be used, the models to be constructed and the evaluation metrics that will be incorporated to perform the comparative analysis.

2.1 Data

The Related Work section of the first chapter included an outline of the legacy dataset to be used in this research, the NSL-KDD. Here a similar analysis on the modern dataset, the CSE-CIC-IDS2018 will be presented.

The CSE-CIC-IDS2018 is a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute of Cybersecurity(CIC) [Sharafaldin *et al.* 2018]. The CSE-CIC-IDS2018 is the most recent network intrusion detection dataset that is publicly available, contains a wide range of attack types and consists of enough network traffic to be considered big data [Leevy and Khoshgoftaar 2020]. It was developed as a comprehensive and diverse benchmark dataset for network intrusion detection, specifically anomaly-based methods [Sharafaldin *et al.* 2018].

The dataset contains 16,233,002 instances of network traffic captured over the course of 10 days [Leevy and Khoshgoftaar 2020]. Seven different attack scenarios are represented in the dataset; Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), Brute-force, Heartbleed, Botnet and inside infiltration [Sharafaldin *et al.* 2018].

A heavy class imbalance exists in the dataset as only about 17% of the instances are attacks. This class imbalance accurately reflects the unequal distribution between benign and attack network traffic observed in real-world environments [Leevy and Khoshgoftaar 2020]. The dataset consists of 80 features extracted from the raw network data by the CICFlowMeter-V3, a network traffic flow analyser [Lashkari *et al.* 2017].

Data pre-processing typically includes one-hot encoding the protocol, service and flag related features as they are non-numerical, categroical features [Gamage and Samarabandu 2020]. Missing values are also removed along with any duplicate records [Basnet *et al.* 2019] and values are scaled using mininum-maximum normalization [D'hooge *et al.* 2020].

As previously noted, a prevelant issue in the CSE-CIC-IDS2018 is the severe class imbalance. Whilst many researchers do not address this class imbalance [Leevy and Khoshgoftaar 2020], Chastikova and Sotnikov [2019] overcome it using the Focal Loss Function while Karatas *et al.* [2020] address it using Synthetic Minority Oversampling Technique (SMOTE). In this research, the class imbalance will be rectified by selecting instances for each class equal to the number of instances in the lowest frequency class; the Web Attack class which appears 96 000 times (0.006% of the total dataset).

Unlike the NSL-KDD dataset which is already split into training and testing data by the providers, allowing researchers to use the respective records accordingly [Yin *et al.* 2017], the CSE-CIC-IDS2018 needs to be partitioned. This will be done using a 70%-30% train-test split of the data as it has shown promising results for NIDS development [Kim *et al.* 2020].

2.2 Models

Deep Neural Network

A Deep Neural Network (DNN) is the basic structure of deep learning networks and comprises of an input layer, multiple hidden layers and an output layer [Jia *et al.* 2019]. Each layer in the network consists of artificial neurons which accept a number of inputs. The product of each input and its respective weight are added together. The combination of the weighted sum of inputs and the bias term is then passed through an activation function to determine the output of the neuron [Lee *et al.* 2018].

The number of neurons in the input layer conventionally matches the number of input features in the dataset whilst the number of nodes in the output layer coincides with the number of target classes in the classification problem [Jia *et al.* 2019]. The number of hidden layers and the number of nodes in each hidden layer is typically determined experimentally, with an increase in model complexity resulting in higher accuracy but also increased training times [Yin *et al.* 2017].

The DNN Model used in the comparative analysis will be developed in lieu of the work done by Jia *et al.* [2019] in which a DNN with 4 hidden layers is used to develop a Network Intrusion Detection System.

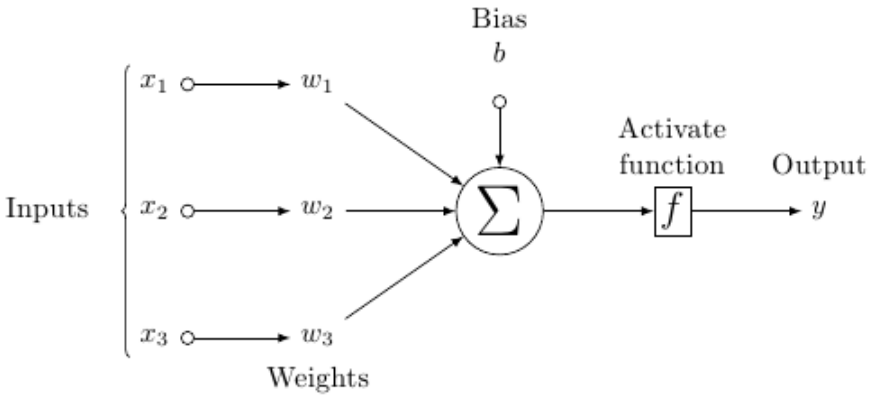

FIGURE 2.1: A single artificial neuron

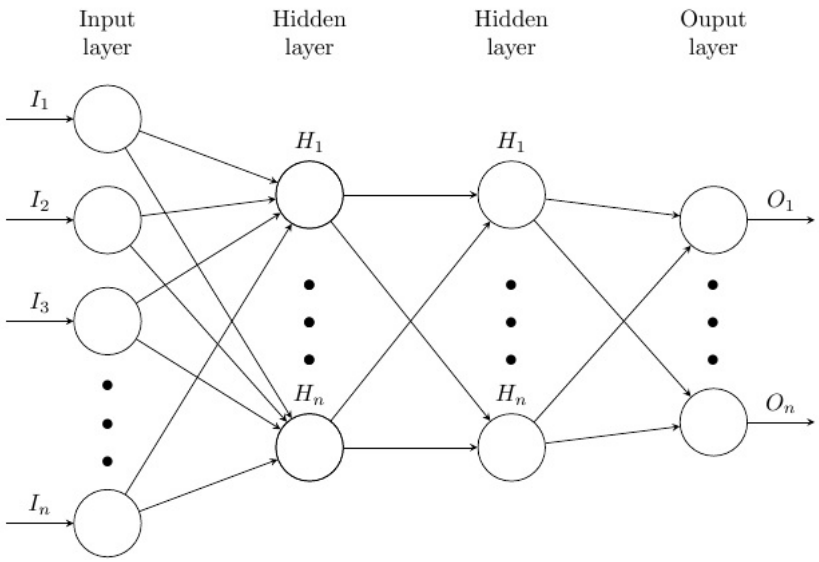

FIGURE 2.2: Deep Neural Network Structure

Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) is the foundation on which Deep Belief Networks are built [Alrawashdeh and Purdy 2016]. An RBM is a generative stochastic neural network consisting of one visible layer and one hidden layer, capable of learning internal representations of data [Ahmad *et al.* 2021]. Each neuron in the visible layer is fully connected to every neuron in the hidden layer with bidirectional, shared weights [Liu and Lang 2019]. An RBM is restricted because no connections exist between neurons in the same layer unlike in traditional Boltzmann Machines [Liu *et al.* 2017].

The training process for RBMs involves a two phase process. In the forward phase of training, the inputs to the visible layer are multiplied by their respective weights and then added to the hidden bias [Hinton 2012]. This combination is then passed through a sigmoid activation function to determine if the hidden neuron gets activated or not [Liu *et al.* 2017]. In the reverse, or reconstruction, phase of training the same process is followed but in the opposite direction. The activation results from the first phase are treated as the inputs to the hidden layer, where they are multiplied by the shared weights and then added to the visible bias [Hinton 2012].

The reconstruction error is the difference between the inputs to the visible layer in the forward pass and the reconstruction output of the reverse pass [Hinton 2012]. Training the model involves adjusting the shared weights of the network, the hidden bias and the visible bias to minimize the reconstruction error [Alrawashdeh and Purdy 2016].

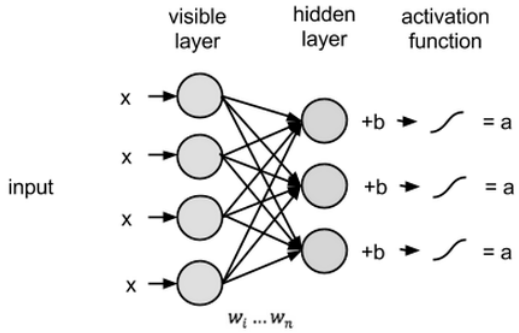

FIGURE 2.3: Forward Phase of RBM Training

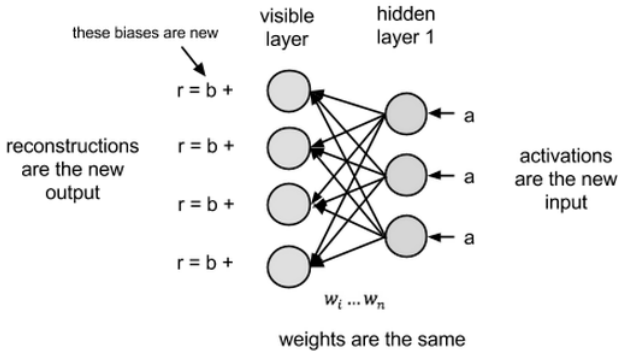

FIGURE 2.4: Reconstruction Phase of RBM Training

Deep Belief Network

A Deep Belief Network (DBN) can be formed by stacking numerous RBMs and connecting the hidden layer of the previous RBM to the visible layer of the next RBM [Liu and Lang 2019]. The final layer consists of a fully connected soft-max layer [Alrawashdeh and Purdy 2016]. In the initial phase of training a DBN, the different RBMs are trained sequentially using unsupervised, greedy, layer-wise pre-training [Liu *et al.* 2017]. In the fine-tuning training phase the network parameters are adjusted further using supervised learning [Liu *et al.* 2017].

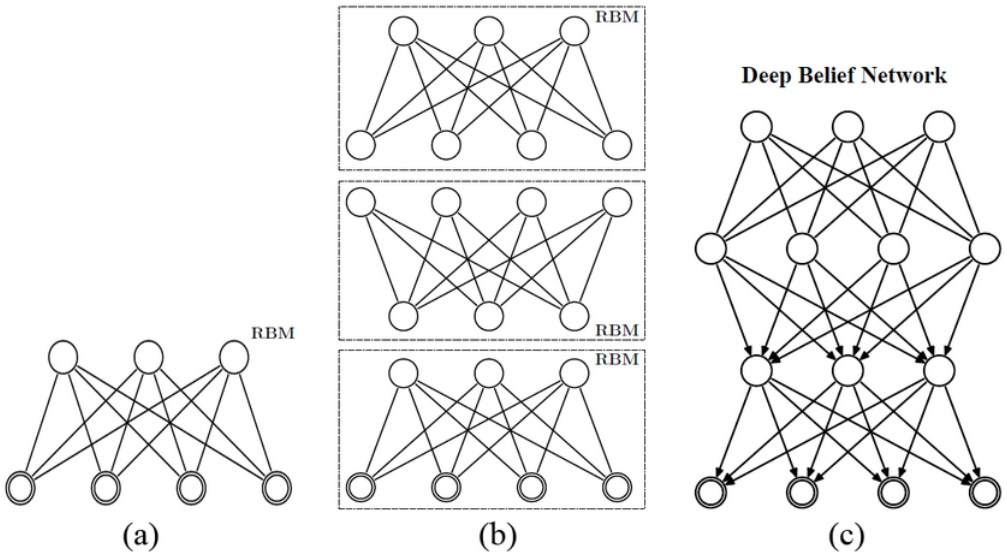

FIGURE 2.5: Progression from RBM to DBN

AutoEncoder

An AutoEncoder (AE) is an unsupervised DL network with one hidden layer that is used to perform dimensionality reduction [Liu *et al.* 2017]. The input layer and output layer of an AE contain the same number of neurons while the hidden layer is usually smaller [Ahmad *et al.* 2021]. First, the encoder between the input and hidden layer performs feature extraction by projecting the input into a lower dimension [Xiao *et al.* 2019]. Then the decoder, between the hidden layer and output layer, attempts to reconstruct the original input using this low-dimensional representation [Yan and Han 2018]. By reducing the dimensionality in this way, the AE is forced to learn the most prominent features of the original data distribution [Shone *et al.* 2018].

Training an AE involves first performing feed-forward propagation to obtain the reconstructed input, then the squared-error between the actual inputs and the reconstructed inputs is back propagated through the network to update the weights [Liu *et al.* 2017]. Once this reconstruction error has been minimized, the decoder is able to reconstruct the input using the features extracted by the encoder [Liu and Lang 2019]. At this point, the output of the encoder can be used as an optimal low-dimensional representation of the original data [Yan and Han 2018].

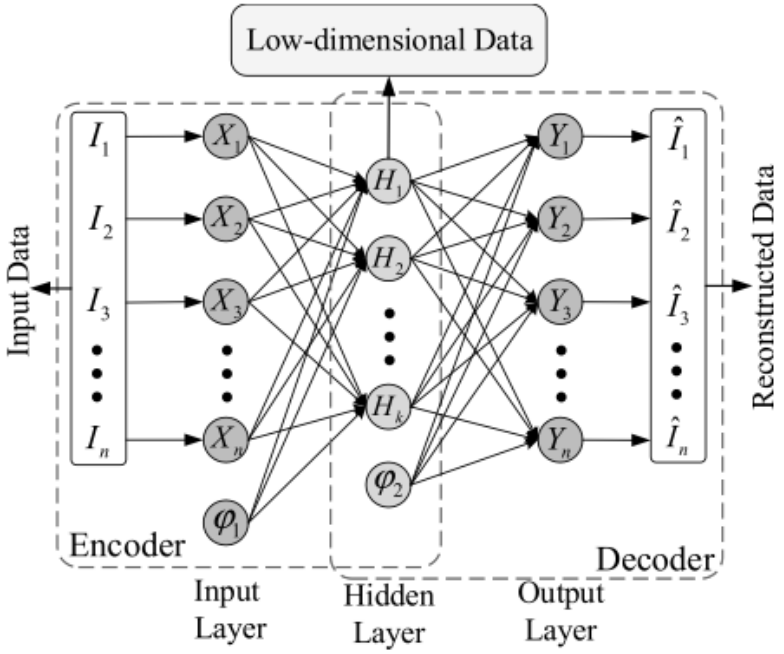

FIGURE 2.6: Structure of an AutoEncoder

Self-Taught Learning

In the literature, a popular approach for including an AE in the development of Network Intrusion Detection Systems is Self-Taught Learning (STL). STL involves two stages, first in the Unsupervised Feature Learning (UFL) stage, a large collection of unlabelled data is used to learn an effective feature representation of the data [Javaid *et al.* 2016]. Then in the second stage, the learnt feature representations are used along with a supervised algorithm to perform the classification task [Al-Qatf *et al.* 2018].

An unavoidable problem that arises when using an AE is that occasionally it will learn to simply copy the inputs directly to the outputs and not extract any meaningful features from the data [Yan and Han 2018]. To overcome this issue, Ng *et al.* proposed the Sparse AutoEncoder (SAE), an AE variant that introduces a sparse penalty to ensure the AE learns a more concise and efficient low-dimensional feature representation [Yan and Han 2018].

The SAE has become a common choice for performing the UFL stage of the STL process as it is relatively easy to implement and boasts a good performance [Raina *et al.* 2007]. In this research the UFL will be done using an SAE while the supervised classification task will be performed using a Soft-Max Regression similar to the model proposed by Javaid *et al.* [2016].

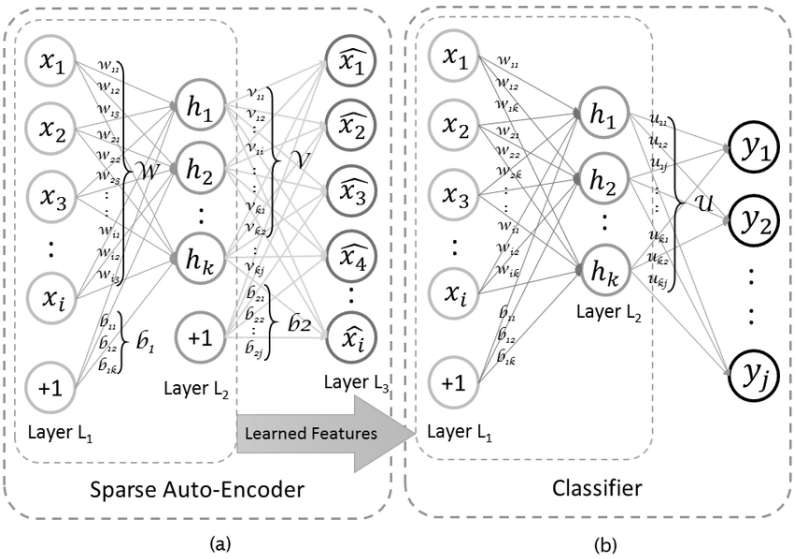

FIGURE 2.7: Two-stages of the Self-Taught Learning Approach. a) Unsupervised Feature Learning. b) Supervised Classification Task

Convolutional Neural Network

A Convolutional Neural Network (CNN) is a DL structure that consists of convolutional and pooling layers to perform feature extraction as well as a fully connected layer to perform the classification task [Ahmad *et al.* 2021].

Convolutional layers create an activation map of the input by sequentially multiplying a matrix of learnable parameters (the kernel) with different section of the input [Goodfellow *et al.* 2016]. Pooling layers reduce the spatial complexity of the representation created by the convolutional layer by deriving a summary statistic for each section of the representation [Goodfellow *et al.* 2016]. Often this summary statistic is just the maximum of each section known as max pooling [Xiao *et al.* 2019].

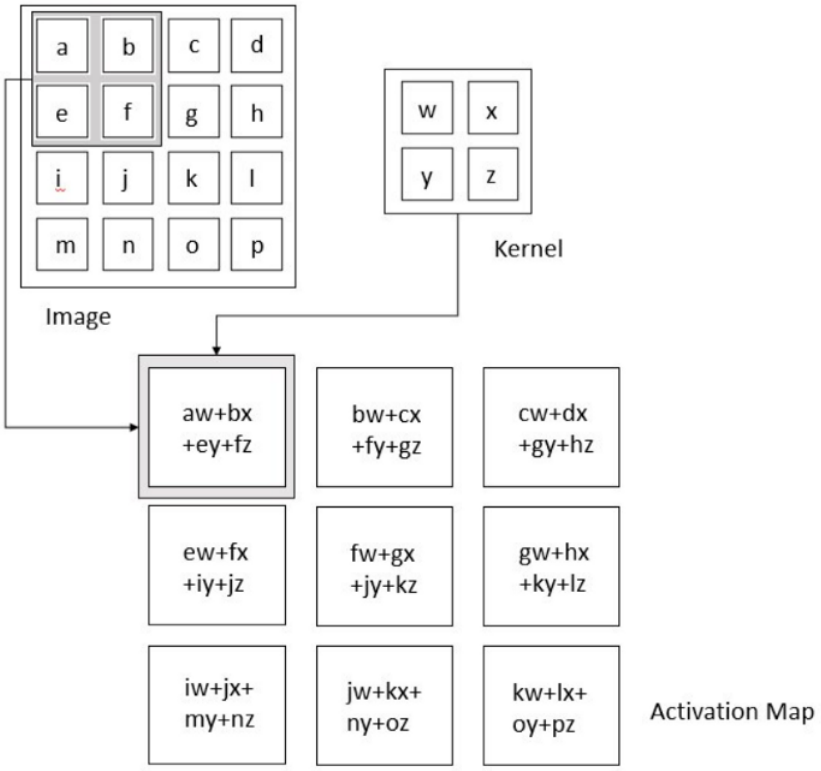

FIGURE 2.8: Convolution operation in Convolution Layer

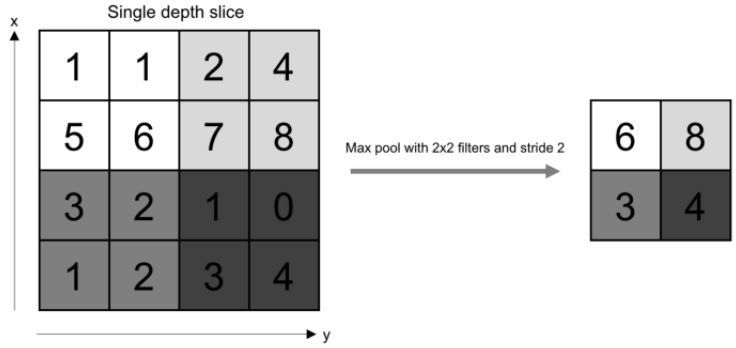

FIGURE 2.9: Pooling operation in Pooling Layer

Training a CNN follows a similar procedure to training a DNN where a forward pass through the entire network is performed and then the error between the observed and actual inputs is backpropagated through the network to update the network parameters [Goodfellow *et al.* 2016].

Due to their reduced number of network parameters and their feature extraction capabilities, CNNs are well suited for the NIDS problem and are able to identify attacks more efficiently than other DL algorithms [Xiao *et al.* 2019]. In order to incorporate a CNN Model into this research, the network traffic first needs to be transformed into the required two-dimensional matrix form thereafter the CNN model will perform the feature extraction and the classification task.

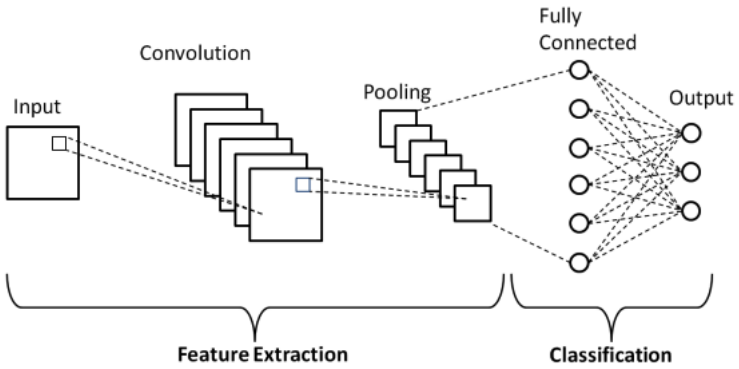

FIGURE 2.10: Convolutional Neural Network Structure

Recurrent Neural Network

A Recurrent Neural Network (RNN) extends the capabilities of traditional DNNs and was designed to model sequential data [Ahmad *et al.* 2021]. The RNN introduces a feedback loop capabable of memorizing previous information and applying it to the current output [Yin *et al.* 2017]. This previous information contextualizes the current state and is incorporated by connecting each neuron in the hidden layer, allowing that neurons output to be calculated using it's respective inputs as well as the output of previous neurons [Liu and Lang 2019].

Training a RNN is once again similar to training a traditional DNN. First, forward propagation calculates the output values and then backpropagation uses the accumulated residuals to adjust the network weights accordingly [Yin *et al.* 2017]. However, the RNN suffers from the vanishing gradient problem where the gradients used in the backpropagation stage become so small that the weights no longer change [Lee *et al.* 2018].

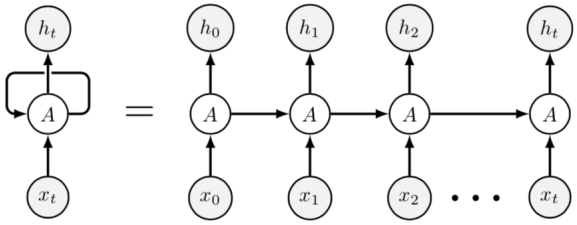

FIGURE 2.11: Recurrent Neural Network Structure

To overcome the vanishing gradient problem, a variant of the RNN known as a Long Short-Term Memory (LSTM) was proposed by Hochreiter and Schmidhuber. The LSTM is capable of learning long term dependencies easily which eliminates the vanishing gradient issue. Each LSTM cell consists of three gates; a forget gate which eliminates outdated memory, an input gate which receives new input and an output gates which integrates both long-term and short-term memory to generate the current output [Liu and Lang 2019] .

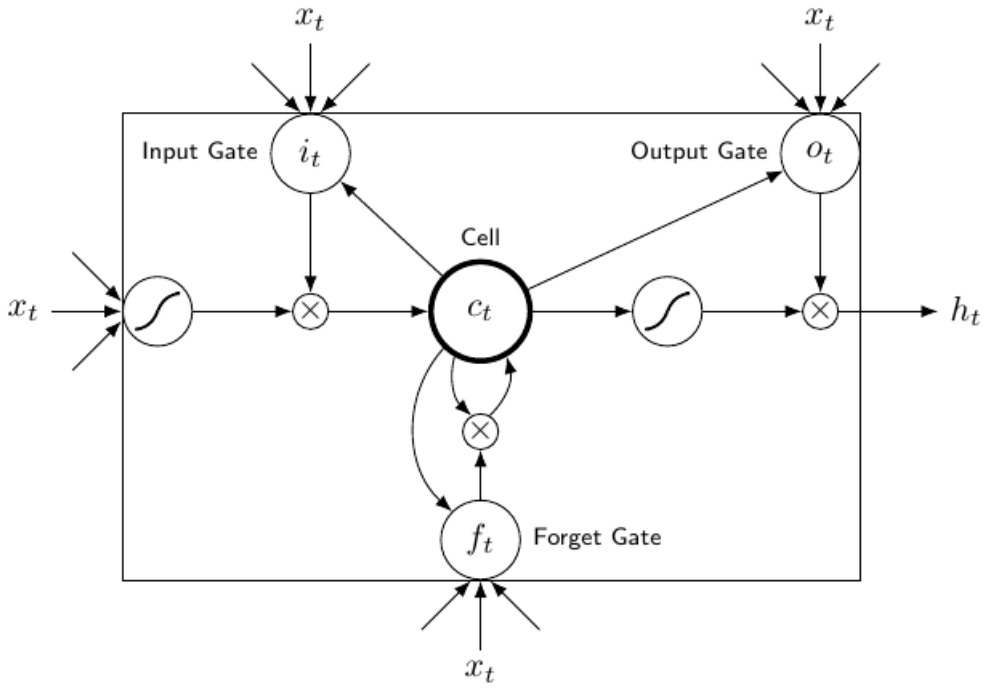

FIGURE 2.12: A Long Short-Term Memory Cell

2.3 Computing System

In order to ensure a fair comparison of the different DL models, each one will be constructed, trained, tested and evaluated on the same computing device using the same software frameworks. Model construction will be done using Python with the TensorFlow and Keras frameworks as these have been used most frequently for recent DL-based NIDS development [Leevy and Khoshgoftaar 2020]. A 13-inch Apple M1 MacBook Pro will be used to carry out the experimental evaluation.

TABLE 2.1: Specifications of the device to be used for the evaluation

13-inch Apple M1 MacBook Pro	
Processor	M1 8-core
Storage	512GB
RAM	8GB
Graphics	Integrated 8-core GPU
Neural Engine	Integrated 16-core

2.4 Analysis

The metrics used when evaluating the performance of a Network Intrusion Detection System are based on the different attributes found in a Confusion Matrix, a two-dimensional matrix which contains information about the actual classes and predicted classes for each instance in the dataset [Ahmad *et al.* 2021]. Each instance is categorized into one of the following four categories [Xiao *et al.* 2019]:

i. *True Positive (TP)* - An attack instance has been correctly classified as an attack
ii. *False Positive (FP)* - A benign instance has been incorrectly classified as an attack
iii. *True Negative (TN)* - A benign instance has been correctly classified as benign
iv. *False Negative (FN)* - An attack instance has been incorrectly classified as benign

A number of useful performance measures can be derived from the Confusion Matrix to better gauge the performance of a NIDS.

		Predicted	
		Attack	Benign
Actual	Attack	*True Positive*	*False Negative*
	Benign	*False Positive*	*True Negative*

TABLE 2.2: Confusion Matrix

- **Accuracy:** Defined as the number of correctly classified instances over the total number of instances, accuracy represents the systems ability to correctly identify benign and attack instances [Hsu and Matsuoka 2020]. Accuracy is the most prevalent performance indicator in NIDS research [Ahmad *et al.* 2021] but is only a good evaluation metric when the dataset is balanced [Xiao *et al.* 2019].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

- **Recall:** Also known as the Detection Rate (DR) or the True Positive Rate (TPR), recall is the number of correctly identified attacks over the total number of attack instances [Shone *et al.* 2018].

$$Recall = Detection\ Rate = True\ Positive\ Rate = \frac{TP}{TP + FN} \tag{2.2}$$

- **Precision:** Represents how many of the instances identified as attacks were actually attacks, it is defined as the number of correctly identified attacks over all instances classified as attacks.[Ahmad *et al.* 2021]

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

- **False Alarm Rate:** Measures the portion of benign instances which were incorrectly classified as attacks [Shone *et al.* 2018].

$$False\ Alarm\ Rate = \frac{FP}{FP + TN} \tag{2.4}$$

The objective of a NIDS is to obtain a high accuracy and detection rate with a low False Alarm Rate [Yin *et al.* 2017]. These three indicators are used most often to evaluate NIDS models [Ahmad *et al.* 2021] and will therefore be the primary evaluation metrics used to empirically evaluate and compare models in this research.

The problem in NIDS development is, however, two-fold. Not only do the performance metrics need to demonstrate that the model is effective but the model also has to be efficient as timeliness is critical in modern NIDS [Shone *et al.* 2018]. Thus, along with the performance metrics outlined above, the comparative analysis will include an examination of the time required to train each of the models on the NSL-KDD and the CSE-CIC-IDS2018 dataset.

2.5 Ethical Considerations

The Code of Ethics and Professional Conduct from the Association for Computing Machinery (ACM) was designed to guide the ethical conduct of all computing professionals and is a useful resource when evaluating the ethical considerations of research [Gotterbarn *et al.* 2018].

Section 1.3 of the code calls on all computing professionals to be honest and trustworthy. This is particularly pertinent when presenting the results of research. Fabricating or making misleading claims about the results obtained would be clear violations of the Code. It is the researcher's responsibility to outline any modification made to the data beforehand and to ensure complete transparency in the model architectures and respective parameters being used.

The researcher should acknowledge any discrepancies in the data being used to carry out the research that could lead to bias in the overall results such as class imbalances, missing data, unrepresentative sampling or erroneous values. Data leakage can also contribute to misleading conclusions being drawn and preventative measures, such as training and testing on separate data, should be employed.

Chapter 3

Schedule of Work

In order to perform a comparative analysis of different Deep Learning-based Network Intrusion Detection systems, each system will have to be constructed, trained and evaluated on both the NSL-KDD and CSE-CIC-IDS2018 datasets. The proposed schedule of work begins with the ingestion and pre-processing of these datasets followed by the development of each of the models outlined in the Research Methodology section above. Once each model has been trained and evaluated, the results will be compiled and interpreted accordingly. The findings will then be documented in a research report and presented at The University of the Witwatersrand's School of Computer Science and Applied Mathematics Innovation Day 2021.

TABLE 3.1: Week-by-week schedule of work

Calendar Week	**Activity**
July 05	NSL-KDD Data Ingestion and Preprocessing
July 12	CSE-CIC-IDS2018 Data Ingestion and Preprocessing
July 19	DNN Model Construction and Training
July 26	DNN Model Testing and Evaluation
August 02	DBN Model Construction and Training
August 09	DBN Model Testing and Evaluation
August 16	STL Model Construction and Training
August 23	STL Model Testing and Evaluation

Calendar Week	Activity
August 30	CNN Model Construction and Training
September 06	CNN Model Testing and Evaluation
September 13	LSTM Model Construction and Training
September 20	LSTM Model Testing and Evaluation
September 27	Finalize Models
October 04	Result Interpretation
October 11	Report Writing
October 18	Report Writing
October 25	Report Writing
November 01	Report Finalizing and Presentation Preparation
November 08	Presentation Preparation
November 15	*Examinations Begin*

Potential Difficulties

The majority of this research will be conducted during the second semester of the academic year, other course-work requirements will become demanding and time will be limited. It is also difficult to predict exactly how long each task will take as some models may take longer than the allocated 2 weeks to construct and evaluate.

In order to accommodate for this unpredictability, the final week of September has been allocated as additional time to finalize the models in-case any of the prior objectives run overtime. The week preceding that is also a dedicated mid-term Research Break in the University's calendar allowing additional time to finish any outstanding work on the models.

Chapter 4

Conclusion

The rapid development and deployment of network-based technologies has led to an increase in the number of cyberattacks that occur. Network Intrusion Detection Systems are a vital cybersecurity tool which aim to identify potential threats and prevent these attacks by monitoring network traffic.

Recently, the incorporation of Deep Learning algorithms into the development of these systems has led to a drastic increase in efficiency. Particularly, the hybrid approach of utilizing Deep Learning to reduce the dimensionality of data through feature extraction, followed by a traditional Machine Learning algorithm to perform the classification task, has gained significant traction in the literature. Alternative approaches which incorporate the temporal aspect of network data have also shown promising results.

These novel systems are, however, being proposed and evaluated using the outdated KDD Cup '99 and NSL-KDD datasets which are not representative of modern network traffic. The utilization of these legacy datasets allows researchers to draw comparisons with other academic literature but leads to unrealistic results and disputable conclusions.

By empirically evaluating five Deep Learning-based Network Intrusion Detection Systems on both a legacy and a modern dataset this research aims to establish a new baseline for future NIDS research and provide deeper insights into the ability of these models to identify network intrusions.

A Deep Neural Network, a Deep Belief Network, a Self-Taught Learning Model, a Convolutional Neural Network and a Long-Short Term Memory Recurrent Neural Network will be constructed using Python with the TensorFlow and Keras frameworks. These models will be trained and tested using both the NSL-KDD and CSE-CIC-IDS2018 datasets. A comparative analysis of the performance of these models will then be carried out using accuracy, detection rate and false alarm rate as the primary evaluation metrics.

Researchers agree that, due to the increasing volume and complexity of network traffic data, Deep Learning techniques will have to be incorporated to develop accurate, real-time Network Intrusion Detection Systems. This research will evaluate how state-of-the-art DL techniques currently perform on modern network traffic data, allowing future researchers to explore different methods of improving these results and propose novel models, using these results as a baseline.

Bibliography

[Ahmad *et al.* 2021] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[Al-Qatf *et al.* 2018] Majjed Al-Qatf, Yu Lasheng, Mohammed Al-Habib, and Kamal Al-Sabahi. Deep learning approach combining sparse autoencoder with svm for network intrusion detection. *IEEE Access*, 6:52843–52856, 2018.

[Alrawashdeh and Purdy 2016] Khaled Alrawashdeh and Carla Purdy. Toward an online anomaly intrusion detection system based on deep learning. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 195–200. IEEE, 2016.

[Basnet *et al.* 2019] Ram B Basnet, Riad Shash, Clayton Johnson, Lucas Walgren, and Tenzin Doleck. Towards detecting and classifying network intrusion traffic using deep learning frameworks. *J. Internet Serv. Inf. Secur.*, 9(4):1–17, 2019.

[Chastikova and Sotnikov 2019] VA Chastikova and VV Sotnikov. Method of analyzing computer traffic based on recurrent neural networks. In *Journal of Physics: Conference Series*, volume 1353, page 012133. IOP Publishing, 2019.

[Dong *et al.* 2019] Yuansheng Dong, Rong Wang, and Juan He. Real-time network intrusion detection system based on deep learning. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 1–4. IEEE, 2019.

[D'hooge *et al.* 2020] Laurens D'hooge, Tim Wauters, Bruno Volckaert, and Filip De Turck. Inter-dataset generalization strength of supervised machine

learning methods for intrusion detection. *Journal of Information Security and Applications*, 54:102564, 2020.

[Elsayed *et al.* 2020] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Ddosnet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 391–396. IEEE, 2020.

[Gamage and Samarabandu 2020] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767, 2020.

[Goodfellow *et al.* 2016] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[Gotterbarn *et al.* 2018] DW Gotterbarn, Bo Brinkman, Catherine Flick, Michael S Kirkpatrick, Keith Miller, Kate Vazansky, and Marty J Wolf. Acm code of ethics and professional conduct. 2018.

[Hinton 2012] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

[Hochreiter and Schmidhuber 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hsu and Matsuoka 2020] Ying-Feng Hsu and Morito Matsuoka. A deep reinforcement learning approach for anomaly network intrusion detection system. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–6. IEEE, 2020.

[Javaid *et al.* 2016] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor 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)*, pages 21–26, 2016.

[Jia *et al.* 2019] Yang Jia, Meng Wang, and Yagang Wang. Network intrusion detection algorithm based on deep neural network. *IET Information Security*, 13(1):48–53, 2019.

[Jordan and Mitchell 2015] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[Karatas *et al.* 2020] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz. Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. *IEEE Access*, 8:32150–32162, 2020.

[Kim *et al.* 2020] Jiyeon Kim, Jiwon Kim, Hyunjung Kim, Minsun Shim, and Eunjung Choi. Cnn-based network intrusion detection against denial-of-service attacks. *Electronics*, 9(6):916, 2020.

[Lashkari *et al.* 2017] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of tor traffic using time based features. In *ICISSp*, pages 253–262, 2017.

[LeCun *et al.* 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[Lee *et al.* 2018] Brian Lee, Sandhya Amaresh, Clifford Green, and Daniel Engels. Comparative study of deep learning models for network intrusion detection. *SMU Data Science Review*, 1(1):8, 2018.

[Leevy and Khoshgoftaar 2020] Joffrey L Leevy and Taghi M Khoshgoftaar. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. *Journal of Big Data*, 7(1):1–19, 2020.

[Liu and Lang 2019] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396, 2019.

[Liu *et al.* 2017] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[Marir *et al.* 2018] Naila Marir, Huiqiang Wang, Guangsheng Feng, Bingyang Li, and Meijuan Jia. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, 6:59657–59671, 2018.

[Raina *et al.* 2007] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007.

[Sharafaldin *et al.* 2018] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp*, pages 108–116, 2018.

[Shone *et al.* 2018] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.

[Suwannalai and Polprasert 2020] Ekachai Suwannalai and Chantri Polprasert. Network intrusion detection systems using adversarial reinforcement learning with deep q-network. In *2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE)*, pages 1–7. IEEE, 2020.

[Wang *et al.* 2017] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *Ieee Access*, 6:1792–1806, 2017.

[Xiao *et al.* 2019] Yihan Xiao, Cheng Xing, Taining Zhang, and Zhongkai Zhao. An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access*, 7:42210–42219, 2019.

[Yan and Han 2018] Binghao Yan and Guodong Han. Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access*, 6:41238–41248, 2018.

[Yin *et al.* 2017] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.