UNIVERSITY OF TURKU

Network Intrusion Detection System using Deep Learning Technique

Cyber Security

Master's Degree Programme in Information and Communication Technology

Department of Computing, Faculty of Technology

Master of Science in Technology Thesis

Author:

Donatus Ifeanyichukwu Edeh

Supervisors:

Ethiopia Nigussie

Antti Hakkala

June 2021

UNIVERSITY OF TURKU

Master of Science in Technology Thesis

Department of Computing, Faculty of Technology
University of Turku

Subject: Cyber Security
Programme: Master's Degree Programme in Information and Communication Technology
Author: Donatus Ifeanyichukwu Edeh
Title: Network Intrusion Detection System using Deep Learning Technique
Number of pages: 75 pages, 2 appendix pages
Date: June 2021

The rise in the usage of the internet in this recent time had led to tremendous development in computer networks with large volumes of information transported daily. This development has generated lots of security threats and privacy concerns on networks and data. To tackle these issues, several protective measures have been developed including the Intrusion Detection Systems (IDSs). IDS plays a major backbone in network security and provides an extra layer of security to other security defence mechanisms in a network. However, existing IDS built on a signature base such as snort and the likes are unable to detect unknown and novel threats. Anomaly detection-based IDSs that use Machine Learning (ML) approaches are not scalable when enormous data are presented, and during modelling, the runtime increases as the dataset size increases which needs high computational resources to fulfil the runtime requirements.

This thesis proposes a Feedforward Deep Neural Network (*FFDNN*) for an intrusion detection system that performs a binary classification on the popular NSL-Knowledge discovery and data mining (*NSL-KDD*) dataset. The model was developed from Keras API integrated into TensorFlow in Google's colaboratory software environment. Three variants of *FFDNNs* were trained using the *NSL-KDD* dataset and the network architecture consisted of two hidden layers with 64 and 32; 32 and 16; 512 and 256 neurons respectively, and each with the *ReLu* activation function. The *sigmoid* activation function for binary classification was used in the output layer and the prediction loss function used was the *binary cross-entropy*. Regularization was set to a *dropout rate* of 0.2 and the *Adam optimizer* was used. The deep neural networks were trained for 16, 20, 20 epochs respectively for batch sizes of 256, 64, and 128. After evaluating the performances of the *FFDNNs* on the training data, the prediction was made on test data, and accuracies of 89%, 84%, and 87% were achieved. The experiment was also conducted on the same training dataset (*NSL-KDD*) using the conventional machine learning algorithms (Random Forest; K-nearest neighbor; Logistic regression; Decision tree; and Naïve Bayes) and predictions of each algorithm on the test data gave different performance accuracies of 81%, 76%, 77%, 77%, 77%, respectively.

The performance results of the FFDNNs were calculated based on some important metrics (FPR, FAR, F1 Measure, Precision), and these were compared to the conventional ML algorithms and the outcome shows that the deep neural networks performed best due to their dense architecture that made it scalable with the large size of the dataset and also offered a faster run time during training in contrast to the slow run time of the Conventional ML. This implies that when the dataset is large and a faster computation is required, then FFDNN is a better choice for best performance accuracy.

Keywords: Cybersecurity; Deep learning; Machine learning; Intrusion detection system; Computer networks

Table of Contents

List of Tables

List of Figures

List of Abbreviation

Acknowledgment

Thanks to Almighty God for wisdom and grace to have achieved this thesis. I also thank my wife and family for their encouragement throughout my studies.

A very big thank you to my supervisors Ethiopia Nigussie and Antti Hakkala for their guardians.

1 Introduction

1.1 Motivation

Computer networks have increased tremendously over the years due to the advent of a digital revolution, and the development has relatively led to a high number of attack vectors on the networks. It is envisaged that by 2023, that there will be an increase in the number of IP-connected devices that would produce an enormous amount of IP traffic up to 4.8ZB [1]. This would pose security challenges to the conventional IDS and traditional machine-learning mechanism already in existence. Internet security threats have continued to rise with an increase in the internet network, and that has made cybersecurity an essential field of research. The major Cybersecurity techniques in use include antivirus software, firewall, and intrusion detection systems (IDSs) [2], which are still playing useful roles today in cybersecurity defence to ensure that networks are protected from internal and external malicious attacks.

In the security of networks, the IDS makes the firewall more progressive by serving as an additional layer to disallow rules based on traffic activities. Business operation depends more on Information technology (IT) and networks. Therefore, the need to provide adequate protection to data is essential. We now live in a digital world, where many critical infrastructures are linked to the internet, and money and information have become digital assets. No doubt, this transition has transformed IT, making it a complex technology with many IoT devices and systems connected to the internet [3].

Spam, Denial of service (DoS) attacks, worms, phishing attacks in one way or the other depends on some form of harmful software called malware. Malware is a programming software designed by cybercriminals to exploit vulnerabilities found in a network of computer systems. Examples include viruses, ransomware, worms, Trojans, spyware, etc.

Over the years, this malicious software has spread and become more complex, most especially from the first decade of the 21st century [4], causing havoc on the computer systems and networks, theft, and other illegal activities, and in this present-day generation has become a serious security issue and have continued to grow. So, there is a need for a stronger network security defence using machine learning techniques that is capable of addressing these security challenges, even up to the future.

Nowadays, the use of machine learning in cybersecurity is becoming popular, and cybersecurity researchers are not relenting in ensuring that a state-of-the-art cybersecurity defence would be achieved, because data has become an essential asset to so many organizations, and because many of these data have been digitalized, they must be given adequate protection in this digital age to preserve integrity, availability and confidentiality of services. The Naïve Bayes (NB), K-Nearest Neighbor (KNN), Random Forest (RF), Logistic Regression (LR), Decision tree (DT), Support Vector Machine (SVM), and Multi-layer perceptron (MLP) are some of the commonly used conventional machine learning methods that have already been applied to anomaly detection and classification, and other areas of cybersecurity such as malware detection. Compared to the signature-based IDS, the machine learning techniques can detect unknown and novel attacks. The challenge faced by these conventional machine learning methods is that they are not scalable to a large dataset because of their shallow architecture, and again their feature extraction phase is done manually.

Machine learning algorithms such as the deep learning approach, have started delivering outstanding results in some major fields of study like medical in area of medicine, diagnosing, treatment, and prevention [5] [6], other areas of applications include self-driving cars, voice search, automatic handwritten generation and automatic machine translation [6].
The deep learning approach would be more sophisticated and with the capabilities to overcome the limitations as found in the traditional machine learning techniques. It is also used in other various areas of cybersecurity applications such as malware detection and classifications, drive-by download attacks, file type identification, spam identification, insider threat detection, network traffic identification, botnet detection, user authentication, false data injection attack detection, verifying human typed keystrokes and border gateway protocol anomaly detection [7].

1.2 Problem

As already discussed in section 1.1, the rapid increase in the number of disruptive technologies such as blockchain, big data, Internet of Things (IoT) has given rise to complex cybersecurity problems due to the amount of new and unknown threats that is been discover daily.
[2] there is usually a high occurrence of high false alarm rate in the conventional IDS which has made it prone to many low non-threatening attacks, and this situation occasionally has led to the ignoring of harmful attacks by a security analyst. Another major issue is that the

traditional IDS cannot detect unknown and novel malware variants such as zero-day attacks as they continue to emerge. This has generated lots of concern to cybersecurity researchers, who have taken the advantage of data-driven engineering to build machine learning algorithms to eliminate these limitations. This is a generation of big data technology and the machine learning approach performance depends on the availability of data it has; however, lack of enough data harms their performances. The use of a Deep Neural Network (DNN) in cybersecurity is a new research interest for many cybersecurity professionals [21], and a deep neural network for IDS is one such research area. This is because the DNNs have the capabilities to manage enormous data that helps them to have a better performance in classification than the traditional machine learning techniques, which previously have shown to be a stronger and better defence mechanism over the signature-based intrusion detection systems.

1.3 Objective

Intrusion detection systems aim to identify intrusion activities and attacks on internet network that is in progress or already occurred, so it is an active security mechanism that is very important, powerful, and a core technology of network security. The goal of this thesis work is to create and analyse a deeper machine learning technique using a deep neural network that can be implemented on a signature or anomalous behaviour of network traffics.

1.4 Research questions

Cybersecurity has become an essential field of research and this thesis work is done to contribute and provide an alternative way that intrusion detection systems (IDSs) can be designed using deep machine learning methods. The focus is based on the discussion in section 1.3 above.

a. What is the importance of using Machine Learning (ML) for cybersecurity?

b. Which ML algorithm is better suited for IDS implementation?

c. How to design intrusion detection systems using the Deep Learning (DL) method?

d. How more desirable is the performance of the designed deep neural network to the existing methods?

1.5 Thesis Organization

The organization of this thesis is as follows:

Chapter one discusses the motivation behind this thesis topic and research interest, problems, objectives, and the research questions to guide towards the completion of this thesis work.

Chapter two is about Machine learning techniques for intrusion detection. It discussed cybersecurity, network security, and network attack types. The different types of intrusion detection methods and intrusion detection systems were also discussed extensively. Another area of interest was machine learning in cybersecurity, how ML has impacted cybersecurity, and the major steps of implementation. It also focused on some commonly used conventional machine learning algorithms (e.g., Support vector machine, Naïve Bayes, Random Forest, Artificial neural network) already applied to building IDS. Deep learning fundamentals, concepts, and the different deep neural network architectures that have been used for cybersecurity were discussed, and finally, a few related works based on deep learning for intrusion detection systems.

Chapter three discuss the design and methods that were employed in this research work. Firstly, the software environments where all experiments were performed, and the tools that supported the success of the experiments were discussed. Also discussed was the description of the NSL-KDD dataset, and the data pre-processing techniques such as feature encoding, feature scaling, and feature selection used for processing the training data and test data. The FFDNN binary and multi-class classifications of the network intrusions, and also the statistical metrics used to evaluate the performance of the models were discussed., Finally, this chapter also focused on the FFDNN model architecture and implementation, and the ML model implementations.

Chapter 4 focused on the experimental process and analysis. discusses the dataset, data pre-processing, model implementation, and analysis. Experimental setup for the FFDNN and Machine learning models, discussion, and analysis based on the simulation result was detailed.

Chapter 5 focused on the conclusion and Future works

2 Machine Learning Methods for Intrusion Detection

2.1 Cybersecurity

We now live in a world of cyberspace where many devices such as the internet of things (IoT), critical infrastructure, network infrastructures, and computer systems are connected to the internet, this has shown that we are now in an era of a digital revolution. A large and sensitive amount of data flow through these networks by the day and need to be protected and treated with so much care and concern to protect them from malicious attackers. Cybersecurity has become the backbone for many businesses, organizations, governments, and individuals to survive in terms of business growth, resources allocation, policy making, data protection, and privacy preservation. It is a major area of interest for security professionals, because of sensitive data that need to be protected.

[8] define cybersecurity as the protection of computers, programs, servers, network infrastructures, and data from unauthorized access or change, by ensuring that the right procedures, policies, and cyber protective measures are applied. Cybersecurity can be categorized into six different areas [3]:

1**. Network security**: This keeps the network and communication of data safe from intrusion with the use of IDS, firewall, and other cyber security defence mechanisms.

2. **Application security**: This security ensures that the applications and software are free of malware infections that can cause data loss and leakage. It can also be seen as the security of web pages.

3. **Information security**: Data is key to many organizations, so the protection of vital information in a database from unauthorized access is of great importance. Data bridge could cause the organization a fortune to deal with.

4. **Operational security**: This security is needed to protect the business daily activities of the organization to ensure smooth running since a large amount of data is handled and transported during this process.

5. **End-user security**: Creating security awareness among the workers is very essential, since they need to understand the fundamentals of cybersecurity and some common cyber threats such as phishing, ransomware, so they don't fall victim to cyber-attacks which will directly affect the security of the organization.

6. **Disaster recovery and business continuity**: This ensures that business operation is returned to normal as soon as possible after a cybersecurity incident had occurred. Inventory of all the organization assets, IT infrastructure, all relevant information, and steps of recovery from a disaster are documented.

The objective of cybersecurity is to ensure the integrity, availability, confidentiality, and nonrepudiation of information and information management systems through various cyber defence techniques [9] [10]. This ensures that information such as medical records, financial records, and personal identity are protected from malicious attacks.

- **Confidentiality**: Ensuring that sensitive information is not exposed to an unauthorized individual and systems.
- **Integrity**: Ensuring that the original information is not tampered with or modified.
- **Availability**: Ensuring that the information is available to the individual or system when it is needed.
- **Nonrepudiation:** Ensuring that information cannot be denied by the sender or system when it is already transmitted.

Today's cybersecurity defence incorporates several defence mechanisms such as firewall, antivirus, network intrusion and spam filter that works in silos to ensure the protection of computer systems and networks.

Cybersecurity defence fights threats at two levels [10], the host-based defence system which protects a host system using antivirus, a firewall, and intrusion detection system, and then the network-based defence system that controls the flow of network traffic also through firewall, spam filter, antivirus, and network intrusion detection system. However, due to the constant increase in the number of threats evolving daily, building defence systems that discover known threats is not enough to protect users, systems, and networks. Cybersecurity professionals, researchers from institutions, private sectors, and government agencies are collaborating, exploiting, and designing different cyber defence systems to address these needs. They have started to involve the use of machine learning (ML) and deep learning (DL) techniques to design and implement safe and secure systems.

2.2 Network security

The major concern of network security is to give individuals the freedom to enjoy computer networks without the doubt and fear that their rights and privacy are in any form of the danger of attack. Therefore, it is necessary to guard the networked computer systems and protect data that are transmitted in the networks or stored in a disk in a networked computer. Again, the internet has become a centre of attraction to many people all over the world and has become a dominant technology. It is built on the IP communication protocol which allows people to communicate and transmit data via a network infrastructure such as a router which can be controlled by other people and is subject to attack. This attraction to the internet has also contributed to the growth in big data technology, thus increasing the amount and complexity of data transmitted from network activities. These activities have grown over the years causing the design of effective IDS to consume high computational cost and overhead, and so many computational resources that can hinder intrusion identification.

Network security is concerned with the protection of computer systems connected to a network from malicious intruders. The goal of network security as already mentioned in *section 2.*1 which is to provide confidentiality, integrity, availability, and nonrepudiation of data that are transmitted or stored in networked computers. So, a major area of focus by cybersecurity researchers is to design network anomaly detection that can detect novel and unknown threats with a minimum false alarm rate [11].

2.2.1 Network Protocols

Networks are organized in a layered manner, where the designer of the upper layer depends on the layer below it. Each layer works based on a predefined services to the layer above it using a predefined Service Access Point (SAPs) utilising the Protocol-Specific Logical Service Primitives to achieve the operation. They are designed to offer a network service that is either connection-oriented, where there is the establishment of connection based on mutual negotiation between entities (sender, receiver, subnet), or a connectionless service that have no logical connection for that data to be regulated as it passes through the network. Networks need protocols to work, these protocols allow efficient and effective communication between a client (web browser) and a server (web server). Network protocols are rules governing the exchange of data between computers connected to the network and can be found in a stack of layers with the layer below communicating with the layer above and verse Versa.

The International Organization for Standardization (ISO) initiated the Open Systems Interconnection (OSI) reference model and designed the computer protocol stack reference model with seven layers (data link, physical, network, transport, session, presentation, and application layers), and also to be used as a framework for the development of protocol standard. Another important reference model called the TCP/IP reference model was also established. Both reference models were combined to form the hybrid reference model to standardize the protocol in Computer networking. To compensate for the limitations on both models, layers 5 and 6 in the OSI reference model were dropped and layers 1and 2 in the TCP/IP reference model were replaced with layers 1, 2 & 3 of the OSI model. This change was necessary because the TCP/IP reference model cannot describe modern networks and does not correctly define the task to be performed in the lowest layer of the model even though it was widely used. On the other hand, the OSI reference model is not widely used even though it can properly describe the network. The majority of the functions provided in the OSI model are also available in the TCP/IP model.

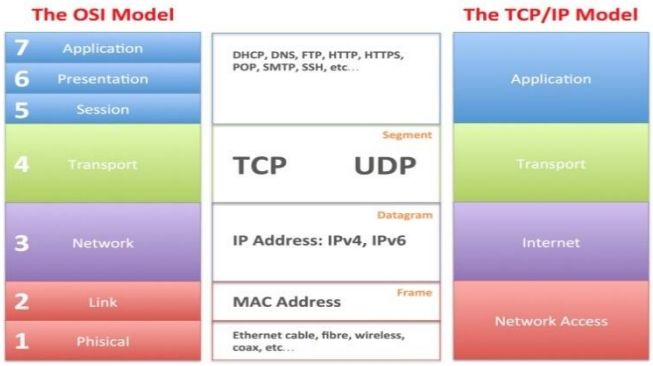

Figure 1: Comparison between the OSI and the TCP/IP models [12]

The layers stacked together in the model in *section 2.2.1* can achieve their functions through service and protocols. The service creates the interface between two adjacent layers, the upper layer is the service provider and the layer beneath is the service user. Again, the service as a set of operations provided by the upper layer is achievable with the help of protocols. The protocols are defined based on the layers, for instance, physical protocol, data link protocol, network protocol, etc. There are different layers with different functions, but to mention are four important layers of the model [13]:

- The *application layer* supports network applications, which are also controlled by other protocols depending on the function of the application. These other protocols are Hyper Text Transport Protocol (HTTP) for web services, Simple Mail Transfer Protocol (SMTP) for electronic mail services, File Transfer Protocol (FTP) that support file transfer.
- The *transport layer* controls the messages from the application layer between a client and a server. Its functions are well coordinated with the help of two important protocols, the Transmission Control Protocol (TCP), which offers a connection-oriented network service, and the User Datagram Protocol (UDP) whose service is connectionless.
- *The network layer* is concerned with the forwarding of packets in a network through a gateway or router. It depends on the services of the link layer, and it supports both the IP protocol and routing protocol which are important for forwarding packets between a source and a destination.
- The *Physical layer* is responsible for the sending of bits of a frame from one node to the other. It is concerned with data encoding, bits representation, and other physical components such as connectors and cables.

2.2.2 Network Attacks

An attack is a chronological succession of illegal events that compromises the security of a network or computer system. [11] classify attacks into seven categories based on implementation.

(i) **Infection attack**: This attack occurs when the target system is infected with malicious files.

(ii) **Exploding**: This occurs when the target system is overflown with malicious code

(iii) **Probe**: The use of software tools to steal or gather information about the target system.

(iv) **Cheat**: Logging into a system with a false identification

(v) **Traverse attack**: This involves trying many different passwords/keys to gain access to a system.

(vi) **Concurrency**: This attack occurs when the service of the system is over-flooded with lots of identical requests that the service is unable to supply.

(vii) **Others**: Attacks in this category take advantage of a vulnerability found on a system.

Table 1: Attack categories with examples [11]

Main Category	Subcategory
Infection	Worms,Viruses, Trojans
Exploding	Buffer Overflow
Probe	Port Mapping Security Scanning, Sniffing
Cheat	MAC Spoofing, Ip Spoofing, DNS Spoofing, Session Hijacking, XSS (Cross-Site Script) Attacks, Hidden Area Operation, and Input Parameter Cheating.
Traverse	Dictionary Attacks, Doorknob Attacks, Brute Force.
Concurrency	DDoS (Distributed Denial of Service), Flooding.

Attacks can also be *passive* in which an intruder monitors the traffic to collect vital information to start an attack, examples are packet sniffing, traffic monitoring, and analysis. Again, it could be *active*, an attack that could cause a devastating effect on a network.
The active attack has been classified into four different categories, namely denial of service (DoS), R2L, probe, and U2R [46] [47].

1. **Denial of service (DoS):** This is a blocking attack whereby an attacker blocks access to the system from legitimate users. In other words, it is an explicit attempt by attackers to prevent the right user access to the service e.g., syn flood, smurf attack, ping of Death.
2. **Probe:** This kind of attack involves gaining information about a remote user from the network. It is deliberately crafted by an attacker on a targeted victim e.g. port scanning using portsweep, IPsweep, Nmap.
3. **Root to Local (R2L)**: Unlawful access from a remote machine, cybercriminal invades into a user remote machine and gains access to vital information e.g. password guessing.
4. **User to Root (U2R)**: Unlawful access to local super user privileges. cybercriminal logs into user account using normal account login and tries to gain administrative privileges by taking advantage of the vulnerability found in the system e.g. buffer overflow attacks.

2.3 Intrusion Detections

It is useful and important to detect abnormal activities by monitoring network traffics that has escaped through the firewalls and steal user and system information, so that system administrators can stop and prevent further damage by the malicious intrusion by taking appropriate actions. This intrusion detection monitoring can be achieved by the use of an automated system called the intrusion detection system (IDS). This idea of intrusion detection was first initiated in the mid-'80s by Dorothy Denning and Peter Neumann [2] [14]. IDSs are essential security mechanisms that play a significant role in network security. It complements other security technology such as firewall, antivirus and access control to offer effective protection to security in today's technology. An intrusion detection system can be a hardware device or software application that monitors a network or a system for suspicious and illicit activities. Over the years many IDS products have emerged as of the writing of this thesis.

The IDS monitors and analyses ingress packets that have bypass the firewall and send alarm signals if any malicious intrusion or attack is detected. There are two major intrusion detection mechanisms according to deployment [11] [17] [18], a *Host-Based Intrusion Detection System* (HIDS) and a *Network-Based Intrusion Detection System* (NIDS). A host-based intrusion detection system is deployed to monitor and analyse system events (application logs, file systems) in the operating system of a host computer. Network-based intrusion detection is deployed through a network device to monitor and analyse network traffics (packets) in real-time. A hybrid detection system will combine both approaches for efficient and effective intrusion detection. The intrusion detection system can further be divided into two types according to the method of detection.

2.3.1 Misuse-Detection Method

This is a signature-based method that relies on detecting the signature of known attacks. It monitors network activities to detect any known attacks that correspond to attacks that are already stored in the IDS. Such monitoring involves searching for known malicious threats by scanning through the network activities, and then it would make a decision based on several important prior knowledge of the attack signature. A generic misuse/ signature detection system comprises five general steps [10], namely data collection, pre-processing the data collected, perform pattern matching and intrusion detection, rule generations, and then defence response. Most times attacks whose signatures are not stored in the system would bypass the

IDS, and this has been the problem of the signature-based IDS. This implies that the technique has the limitation of detecting unknown attacks, but the good is that it has a high rate of detection and the alarm rate is low, and can easily be implemented. Another important aspect of this method is that it can be used to detect and recognize unique patterns (patterns from network packets and log files identified as threats) of suspicious and unauthorized behaviour which can be used against any further attacks of such similar behaviour.

Another drawback of this method is that it is designed based on domain expert's knowledge which varies in terms of experience, and as such so much anomalous behaviour might not be covered which can lead to inadequate detections.

2.3.2 **Anomaly Detection Method**

This method combines machine learning and statistical approach to recognize normal network traffic differently from abnormal traffic. Its goal is to monitor the network activities to find behaviour that is unusual (anomaly behaviour) or that does not conform to the normal behaviour of network activities. It has the advantage of detecting new attacks based on its approach and can be classified based on three different styles of anomaly detection [11]:

- **Supervised anomaly detection**: A model is designed by training a dataset that has instances for normal class and anomaly class. The designed model then predicts unseen data to determine which class it belongs to.
- **Semi-supervised anomaly detection**: The model design is similar to that of supervised anomaly detection, but the training data has labelled instances for only the normal class, and no labels for the anomaly class.
- **Unsupervised anomaly detection**: The designed model does not require training data, instead it assumes that normal instances occur much frequently than anomaly instances in the test data. This technique suffers from a high false alarm rate once the assumption fails.

For the network anomaly, anomalies occur due to [11] (i) anomalies from network operations; (ii) When servers are unable to respond to many requests within a given time, this is called flash crowd; (iii) Anomalous from malicious activities in a network, this can be described from three points of view. It is a point anomaly when there is a deviation of event from previous normal activities. It can be a contextual anomaly that describes an instance that is exceptional within a

given context, and finally, it could be a collective anomaly where a group of instances deviates abnormally with reference to a given normal behaviour of activities in a network.

Table 2: Differences between Misuse detection and Anomaly detection [2]

	Misuse Detection	**Anomaly Detection**
Detection performance	Low false alarm rate; High missed alarm rate	Low missed alarm rate; High false alarm rate
Detection efficiency	High, decrease with scale of signature database	Dependent on model complexity
Dependence on domain knowledge	Almost all detections depend on domain knowledge	Low, only the feature design depends on domain knowledge
Interpretation	Design based on domain knowledge, strong interpretative ability	Outputs only detection results, weak interpretative ability
Unknown attack detection	Only detects known attacks	Detects known and unknown attacks

2.3.3 Hybrid Detection Method

This combines the capabilities of misuse detection and anomaly detection systems to improve the techniques of the IDSs. This implies that the hybrid detection system can detect known threats and also unknown threats. Researchers believe that this development will solve the issues of the drawbacks found in the misuse system of being unable to detect unknown intrusions, and the anomaly detection producing a high percentage of false alarms. The outputs gathered from the anomaly and signature-based methods by the HDM are used to make a final decision on the probability of an attack. Ozgur et al [15] proposed a hybrid detection system that combined both anomaly and misuse detection approaches, also included a decision support system that will manage the outcomes of both detection approaches. They used the Self-Organizing Map (SOM) structure to model normal behaviour in the anomaly detection method and then used the J.48 decision tree technique to classify different attacks in the misuse method. Any deviation from normal behaviour is considered an attack.

2.3.4 Stateful Protocol Analysis Method

This performs the same manner as the anomaly-based method, but it identifies deviations of protocol state in a state table. It gathers information about connections between the host and remote computer and compares it to entries in the state table. The information includes source IP port and Address, destination IP port and address, and the protocols in use. Other functions perform by this IDS are, protocol state tracking, traffic rate monitoring, IP packet reassembly and dynamic application layer protocol analysis [16].

Table 3 Types of Intrusion Detection Approaches [17]

Method	**Pros**	**Cons**
Signature-based IDS	Simplest and effective method to detect known attacks	Ineffective to detect unknown attacks
Anomaly-based IDS	Effective to detect unforeseen vulnerabilities	Anomaly is not always an indicator of intrusions and may increase false-positive rate
Hybrid Approach	Reduce the false positive rate of unknown attacks	Model might be complex
Stateful Protocol Analysis	Know and trace the protocol; state	Unable to inspect attacks looking like benign protocol behaviours.

2.4 Machine Learning in Cybersecurity

Due to the advent of big data technology in this 21st century, there has been an enormous amount of data distribution in the network which could be vulnerable to attack. Therefore, machine learning and statistics, coupled with some other relevant inter disciplines are needed to solve the challenges of cybercrimes.

Machine learning is a subset of artificial intelligence that makes the computer learn from data [17] combining different disciplines as statistics, data mining, and data science. Models can be built from machine learning algorithms and such models can be used to predict newly input data. Machine learning can be divided into two categories *shallow learning* and *deep learning* [9]. *Shallow learning* was the traditionally used method, also called conventional machine learning methods that are based on the learning of data without using networking, they are fast and mostly perform very well on a small amount of data. They are still in use in today's

cybersecurity because they have been used and found to be very useful and effective in different areas of cyber defence [3]. On the other hand, *deep learning* methods are advanced and capable of handling a large amount of data. It automatically extracts relevant information from the data which are necessary for building a system.

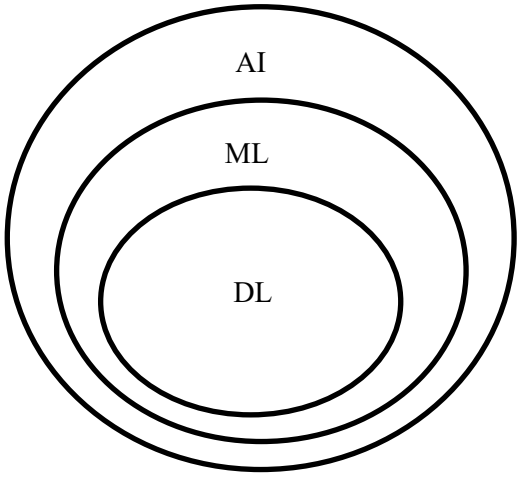

Figure 2: Relationship between AI, ML, and DL

ML technique can be automated to analyse threats, attacks, and other security incidents quickly and efficiently. It can be deployed for a variety of cybersecurity problems such as DDoS attack detection, malware detection, spam mail, phishing detection, user identification, social media analytics, detection of software vulnerabilities, detection of advanced persistent threats, detection of information leakage, detection of identity theft and anomaly detection [19].

Machine learning algorithms are very useful for improving the detection performance of a system [3], the algorithm study and learn the patterns in the data fed into it and then use the trained data to prevent data breaches that could harm the system in the future. Generally, machine learning algorithms are designed to deal with three kinds of problems, classification, regression, and clustering and the learning involve three general phases, pre-processing, training, and detection [9].

- **Pre-processing:** This involves the collection of data from the network environment. Feature engineering and feature selection are performed on the data before they are fed into the machine learning algorithm.
- **Training:** The algorithm trains the pre-processed data and learns the unique characteristics present in the different types of data, and then builds a model based on these characteristics.

- **Detection:** The system model built from training is used to compare to the network traffic to be monitored. If there is a deviation or anomaly in the observed pattern, pattern been observed matched an existing threat, the model triggers an alarm.

2.5 Machine Learning in Network Intrusion Detection

Machine learning methods are vital to the building of IDS, these methods could be identified in three different paradigms*, supervised unsupervised* and *hybrid* methods. Network intrusion detection is considered a classification problem, that requires a labelled training dataset for modelling. In most cases the labelled data that correspond to normal behaviour is available, but label data for anomaly detection are not [10]. To build a machine learning algorithm that is efficient, attack-free training data is required, but it is difficult to obtain such data in a real-world network [10], as such causes an imbalance of data distribution in the design of IDSs.
Machine learning approaches that have been used for IDS include *supervised learning methods* such as the Artificial Neural Network (ANN), Decision tree (DT), K-nearest Neighbour (KNN), Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Naïve Bayes (NB) [9], *unsupervised learning methods* such as the K-means clustering, Self-Organized Map (SOM), and the *hybrid methods* [2] [9]. They have produced good results, though their performances are limited due to the shallow architecture that has made them not scalable to large datasets, and their feature extraction is not automatic.

2.5.1 Decision Tree (DT)

Ethem Alpaydin [20] defines a decision tree as *a "hierarchical data structure implementing the divide-and-conquer strategy"*. It is a nonparametric technique that is suitable for both classification and regression problems. The model is tree-like in architecture which can easily be interpreted. It learns by performing feature selection, generating, and pruning trees. During training, the algorithm can select the most suitable features and build child nodes from the root node. Decision tree classifiers that have been used are ID4, C4.5, and CART, they have better accuracy for known intrusions but are not good at detecting unknown intrusions [21].

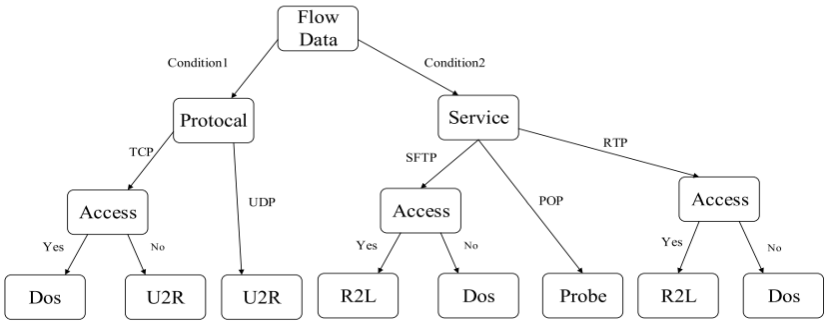

Figure 3: Example of a Decision tree with attack classification [21]

Ingre et al [22] proposed a decision tree-based IDS for the NSL-KDD dataset. 14 features were selected from the dataset using the correlated feature selection (CSF) method. The overall accuracy was 83.7% and 90.3%. Azad et al [23] also proposed an intrusion detection system based on a C4.5 decision tree on a KDD Cup 99 dataset and a high accuracy of 99.89% was achieved.

2.5.2 Logistic Regression (LR)

This is a logarithm linear model mostly used to model dichotomous outcome variables (binary). It is used to describe data and to explain the correlation between one dependant binary variable and one or more nominal or ordinal variables. It is efficient for training a model and easy to construct, and Its probability can be calculated through a parametric logistic distribution [2].

$$P(Y = k/x) = \frac{e^{w_k * x}}{1+\sum_{k}^{k-1} e^{w_k * x}} \quad (1)$$

Where k = 1, 2, … k-1, and x is classified into the maximum probability class

Kamarudin et al [24] proposed an anomaly-based detection model using a binary logistic regression technique combined with a statistical method. The model was designed to detect R2L and U2R attacks at different instances by examining the degree of normal field values within the data link layer, network layer, and transport layer of the OSI Seven Layer Model. The result obtained outperformed other existing methods (DARPA best system, PbPHAD) at 78.95% for R2L and 84.61% for U2R.

2.5.3 Support Vector Machine (SVM)

This is a maximum margin method that allows a model to be written as the sum of influences of some of the features of the data (or of some of the training data points). These influences can be calculated through sophisticated kernel functions, that allow the building of complex models using only linear formalities. The SVM aims to find the best hyperplane that divides classes by maximizing the distance between the hyperplane and the nearest samples of each class (support vector).

Gao et al [25] In 2009 has proposed an intrusion detection approach based on classifying SVM. They used the method of genetic algorithm to optimize the SVM parameters to help improve the detection accuracy and rate of convergence. The model performed well with the mean squared errors of train sample and test sample of 0.0047 and 0.0596 respectively. Recently, this year 2021, Jiang et al [26] had proposed a new intrusion detection method based on an improved SVM named Class and Sample weighted C-support Vector Machine (CSWC-SVM). SVM in the recent past has been successful as a classification algorithm in many classification problems because of its high generalizing performance and global optimal convergence [26].

2.5.4 K-Nearest Neighbor (KNN)

K-Nearest Neighbor classification algorithms make predictions depending on the k nearest samples in the feature space after memorizing the training data. It is a non-parametric, lazy learning algorithm that performs classification based on a distance function that measures the difference or similarity between two instances in the dataset. Considering x and y as the two instances, the standard Euclidean distance $d(x, y)$ between them can be calculated as

$$d(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \quad (2)$$

Where x_k and y_k are the kth featured elements of instance x and y respectively, and n is the number of features present in the dataset [3].

Atefi et al [27] used the CICIDS-2017 dataset to perform anomaly analysis of the intrusion detection system for classification using the K-Nearest Neighbors and Deep Neural Network (DNN). The idea was basically to improve on the detection accuracy of IDS using the newest dataset as against the old dataset of Kddcup'99. The KNN model performed 88% and the DNN performed 92% detection accuracies.

2.5.5 Naïve Bayes (NB)

This is a classifier based on the theory of probability. It applies Bayes theorem with independent attribute guess to computes the conditional probabilities of the data of a given class and the prior probabilities of each class during training. This implies it takes to rely on the conditional probability and the hypothesis of attribute independence [2]. Let's assume the features are independent statistically, then the probability of sample S represented with a set of features 'a' associated with a class C can be computed [28]

$$P_r(c/S) = P(c) \times \pi_{i-1}^{d} P_r(a_i/c) \tag{3}$$

B. S Sharmila et al [29] performed an experimental comparison of Naïve Bayes algorithm and Principal Component Analysis (PCA) on IDS-based implementation using Scikit learn python library. According to their experimental result, the PCA-based NSL-KDD IDS had better accuracy than the Naïve Bayes IDS.

2.5.6 Random Forest (RF)

Random Forests are a group of decision trees constructed at training time for classification or regression tasks, and each tree has been trained on different subsets of the training data. During training, a bootstrapped dataset for a set of decision trees is created and prediction is made based on the highest number of votes. They have the advantage of less overfitting and generalizing better.

J. Zhang et al [30] proposed a new systematic framework using Random forests in misuse, anomaly, and hybrid network-based IDS. The misuse random forest classifier was built based on pattern intrusion and was able to detect intrusions when matched with the network activities. The anomaly-based random forests were able to detect novel intrusions, and the hybrid combined the advantages of both the misuse and anomaly to improve the performance of the IDS. Another recent discovery using random forest classify was presented by Guowei ZHU et al [31], they proposed a power system network intrusion detection method based on a random forest algorithm. The random forest decision tree was constructed from the power system network intrusion sub-sample, and the Gaussian mixture clustering was used to process the training dataset into different clusters, and an RF classifier was trained on each cluster. The accuracy was obtained by calculating the measurement residual of the power system network attack, and the experimental result shows that their proposed system has high network intrusion detection performance.

2.5.7 Artificial neural network

The idea of an artificial neural network was inspired by the biological neuron of the human nervous system and brain. Biologists, however, have shown that the human body system comprises thousands to millions of nerve cells called neurons. These neurons are interconnected together in a very complex manner to perform their various functions in the human body system. The idea was adopted to create the first neuron called the McCulloch-Pitts model in the 1940s [32]. Over the years, several models of the artificial neural network have been developed based on different topologies, learning algorithms, and functionalities.

Yusuf Sani et al [33] in 2009 have published their paper in IEEE. They discussed the use of neural networks in anomaly intrusion detection systems. The feedforward network and recurrent neural network were used as a case study. The paper explains how the neural network IDs should be implemented to overcome limitations (building so many signatures relative to different attacks, cannot detect zero-day attack) found in existing IDS. They believe that neural network-based IDS have better advantages such as better performance, less development cost, highly scalable, and can reduce false positive and false negative error rates. This paper is one of the several papers published as regards the importance of artificial neural networks for IDS in the previous years.

2.5.7.1 The Architecture of Artificial Neural Network

There are several nodes interconnected together according to specific network architecture in the neural network, and each node receives an input with a signal strength called the weight. Every Artificial neural network is made up of the input layer, hidden layer, and output layer.

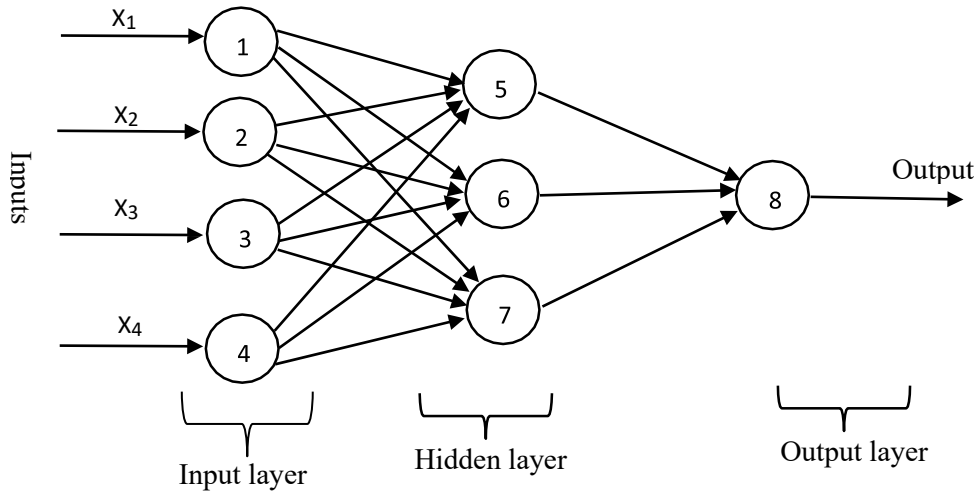

Figure 4: A Simple Structure of an Artificial Neural Network

The Components of the artificial neural network consist of three major components:

- **Weights:** This determines the strength of the inputs, for a network with several neurons, the weights w_i are represented in a vector space, which multiplies an input vector x_i of the same dimension.

  Considering the weight vector as $w_i = [w_1\ w_2\ w_3\ w_4\ w_5]^T$ and

  Input vector, $x_i = [x_1\ x_2\ x_3\ x_4\ x_5]$

  Therefore, the strength of the input signal becomes:

  $$w_i x_i = [w_1\ w_2\ w_3\ w_4\ w_5]^T\ [x_1\ x_2\ x_3\ x_4\ x_5] \quad (4)$$

- **Thresholds:** This is a value that is uniquely assigned to a neuron, it marks the position of the maximum gradient value of the activation function. Neurons get activated when the input signal exceeds a threshold value.
- **Activation functions:** This function is also called the transfer function, threshold function, or squashing function. It determines the activation of a neuron depending on the network input and threshold value, it takes the combined inputs, applies a function on it, and passes out the output values. It depends on the previous activation state of the neuron and the external input. There are different activation functions depending on the kind of problem to be solved, but the three common choices are Sigmoid, ReLU, and Softmax.

  Other components include the *input layer* which contains a certain number of neurons that should be equal to the dimensionality of the data. It takes inputs and distributes them to the *hidden layer*. The number of units in the *output layer* is a function of the type of problem to be solved, whether it is a binary classification problem or multiclass. The *inner layers* in the network are the hidden layers that define the depth of the neural network, and within the hidden layers, complex computations occur.

2.6 Deep Learning Methods and Concepts

Deep learning has become an increasing area of demand to many researchers in different fields most especially in science and technology. Models created from deep learning can be applied to many different tasks in cybersecurity, finances and stock market, medicine, image processing, search engine, and pattern recognition [34]. This sudden growth of DL could be attributed to a few factors such as the increase in the amount of data (especially unstructured data) as a result of the digital age that has made the number of devices connected to the web

increase. Improvement in computing infrastructure is another contributing factor. Deep learning programming can be implemented much faster with the advent of computing layers such as Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) at affordable cost. The sudden rise in the growth of DL was also initiated by the availability of different open-source frameworks (TensorFlow, Keras, PyTorch, MXNet, and Caffe) that could be used to build and implement the DL models. The DL models were able to offer extreme versatility and performance in the different areas they were applied to [2].

Deep learning is a branch of machine learning that deals with different types of artificial neural networks with more than one hidden layer. A conventional artificial neural network already discussed in *section 2.5* consists of only one hidden layer. This is a shallow neural network that does not possess the power to perform feature extraction [17] and is not dense enough. The more hidden layers in the architecture of an artificial neural network, the deeper the neural network becomes, thus acquiring the name Deep Neural Network (DNN). Deep learning models can learn features from input data, that is, learning different levels of features across several layers of its network. More so, they can scale well on a large dataset to gain better results in terms of performance [35], meaning the model's performance gets better with more data.

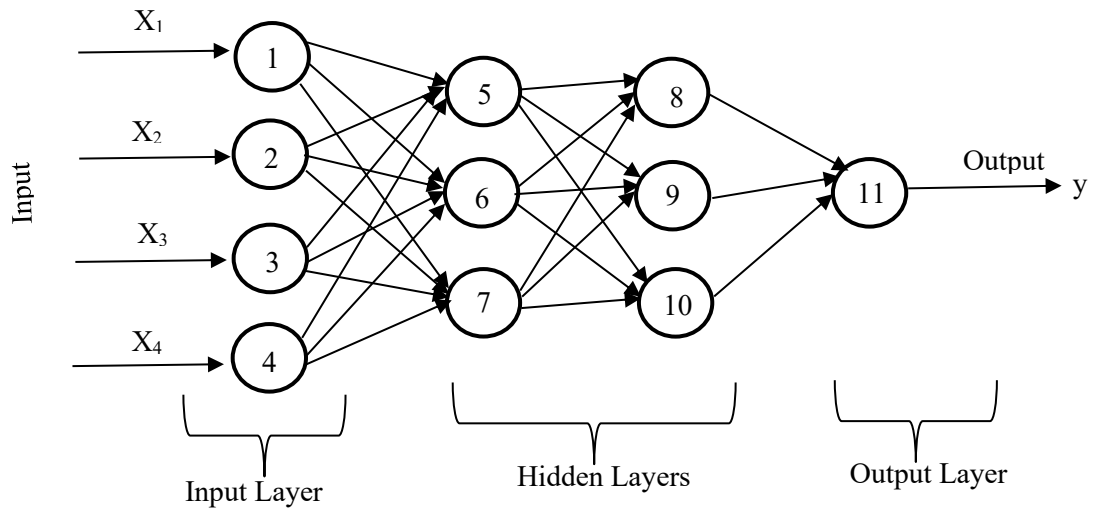

Figure 5: A Deep Neural Network Architecture

There are different deep learning methods based on diverse different architecture [2][17] and can be classified into three groups, namely, generative, discriminative, and hybrid DL models:

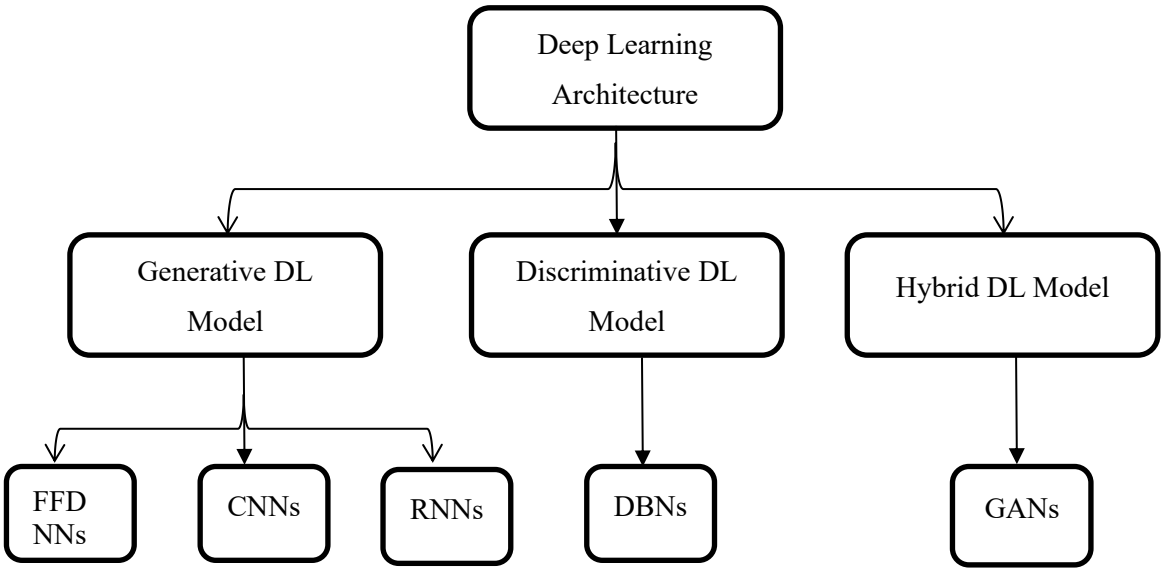

Figure 6: Different Deep Neural Network Architectures

- **Feedforward Deep Neural Networks (FFDNN)**

  The feedforward neural network is a multilayer neural network that has only one direction, from the input to the output, and can be trained through backpropagation. The FFDNN architecture is obtained by increasing the number of hidden layers in the architecture, thus making it a deep neural network.

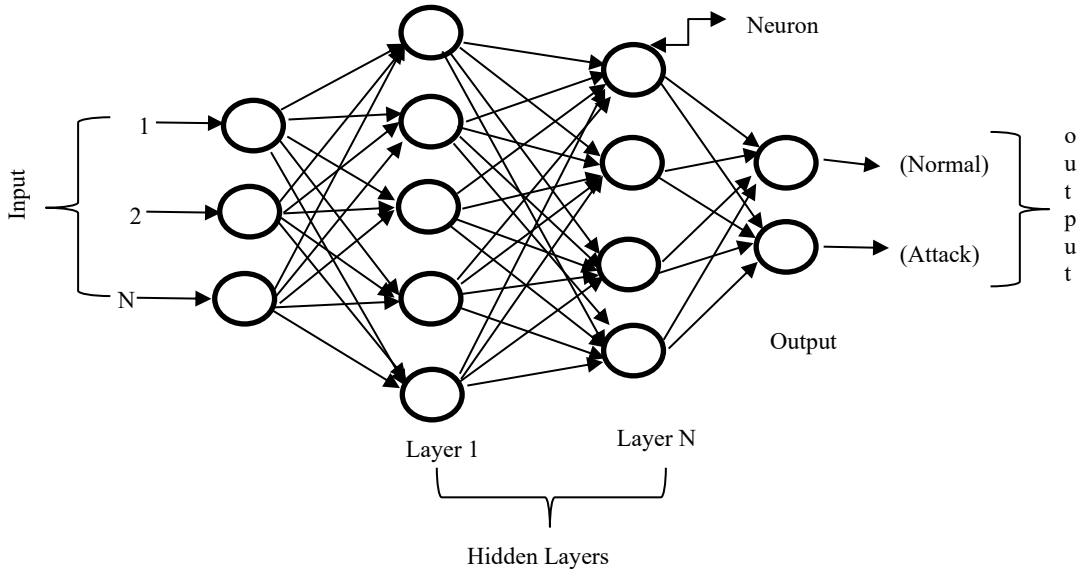

Figure 7: Representation of layers of the FFDNN

- **Convolutional Neural Networks (CNN)**

  This process is input stored in arrays [17], like colour or grayscale images that are 2D of pixels, three-dimensional (3D) arrays (e.g videos), and one-dimensional(1D) arrays (e.g signals). It is best fitted for data that are spatial (unstructured). In 2012, CNN recorded success at the ImageNet competition, with an outstanding performance over other methods [36]. It has been helpful in several other areas such as language modelling (speech recognition), computer vision, image detection, facial recognition, medicine, and cybersecurity.

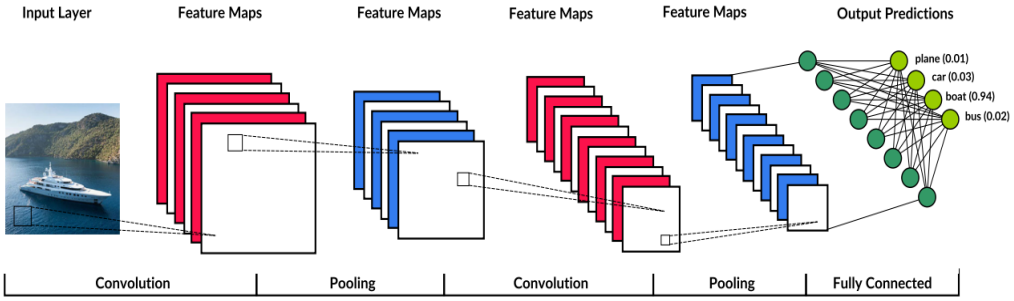

Figure 8: Overall Architecture of CNN for a Classification task [17]

CNN is made up of three important layers, the *convolution layer*, *pooling,* and *classification* layer. The convolution layer is the core part of the CNN where the data processing begins. it is used to extract important features from the image by performing two distinct steps, feature detecting (making matrix or pattern on the data to transform it into a feature map) and feature mapping (Obtaining small images from feature detecting), this process is called *convolution operation*. The work of the pooling layer is to reduce the dimensions of the feature maps so that only the relevant features are kept. This will also help to reduce the computational time of the pooling operation. This process can also be called *downsampling* or *subsampling*, it has three different types, *max-pooling*, *average pooling,* and *sum pooling* [37]. The fully connected layer is a feedforward neural network that performs classification on the extracted features.

- **Recurrent Neural Network (RNN)**

  RNN is a special type of feedforward neural network that processes data in a sequence of times. The design of its network is based on the concept of sequential memory [38]. The hidden layers, after receiving information and giving output, still feedback the information of the output into itself, that is, data and information flow in a cycle. This

makes it a robust technique for processing sequential data, which is not possible in another deep neural network (CNN, FFDNN). RNN can retain information in its memory, but it cannot store this information for too long when the input sequence becomes too long, at this point, it begins to have the problem of vanishing gradient. This issue of vanishing gradient was resolve by the design of another type of RNN called the *Long short-term memory* (LSTM). This variant is made up of a special structure called the 'gate', this will keep the information in the memory for a longer period as required, and can also discard information when it is no longer needed. The deep neural network has been successfully used for speech recognition, language translation & other time series prediction task [17].

- **Deep Belief Network (DBN)**

  This is a class of unsupervised deep neural networks adapted to different functions. It is made up of multiple layers of hidden neurons. The layers are connected, but there is no connection between the hidden neurons. It comes in three different variants [17], Deep Autoencoders, Restricted Boltzmann Machines (RBM), and in combined form (DBNs or RBM, or Deep Autoencoder coupled with classification layers). Each variant is designed to solve specific kinds of problems, and they have been used in the area of cybersecurity for intrusion detection systems.

- **Generative Adversarial Networks (GAN)**

  This deep learning algorithm was introduced by Ian Goodfellow et al in 2014 [39]. It is an unsupervised deep neural network that combines the architecture of two neural networks. The two neural networks compete against each other to optimize weights and biases with the purpose to minimize their errors. One neural network act as a generator that takes input data and produces output data (fake data) which have the same attributes as the real data, the second neural network acts as a discriminator and takes the fake data and real data, and tries to distinguish between the two. After the training, the generator must have learned and is capable of generating new data that is not distinguishable from the real data.

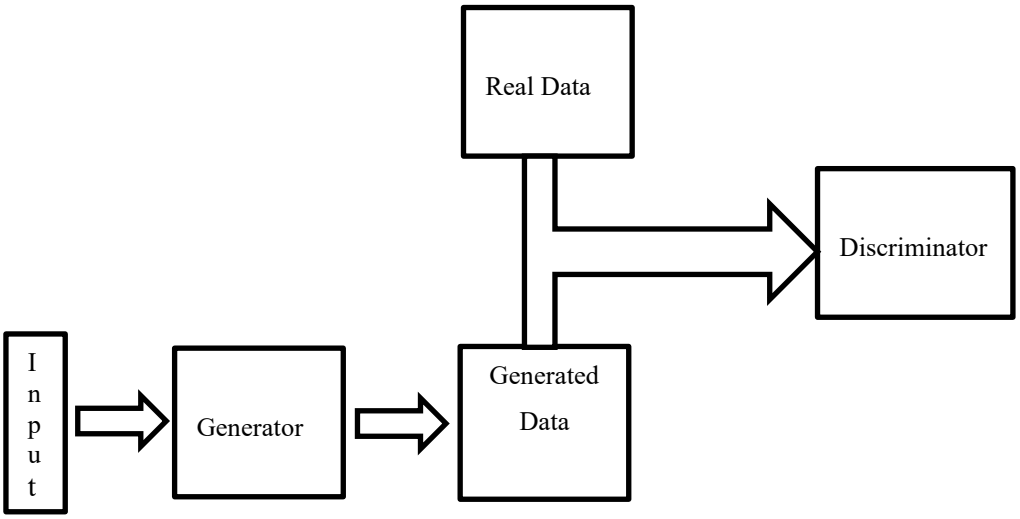

Figure 9: Generative adversarial network [7]

GAN has been used for Image creation such as image enhancement, caption generator, and optical flow estimation [17].

2.6.1 Terms Associated with Deep Neural Networks

(a) Loss function

Training the samples and calculating the output on the training inputs, and then comparing the result with the real label defines the error function or loss function. So, there is a need to define a function that measures error when training a model. Defining the loss function for a deep neural network depends on the problem and objectives to be achieved because different networks have different predictions based on the inputs. For classification, the loss is obtained by computing the probability of the model error which is also the proportion of misclassified inputs in the dataset. Two common loss functions used for classification are:

1. Binary cross-entropy: This is used when dealing with two-class/binary classification problems. The output is a probability between 0 and 1.

2. Categorical cross-entropy: This is defined for a multi-class (more than two classes) classification problem.

(b) **Activation function**

Section 2.5.7 already discussed the activation function. When designing a deep neural network, aside from choosing the size of the layers and number of neurons, the activation function is also an important parameter that needs to be chosen for the hidden layers and output layer. The activation function is chosen depending on the nature of the problem to be solved and how well the performance of the model is evaluated. Some of the ones mentioned below were used in this research work.

- **Sigmoid**

  This activation function is used at the output layer for a binary classification. Its output is the probability of a given input belonging to a class. The out of a sigmoid function is between 0 and 1. Mathematically, it can be expressed [18] as:

$$Sigmoid = \frac{1}{1+e^{-x}} \tag{5}$$

- **Tangent (Tanh)**

  The output of the tanh function falls in the range -1 and 1. It is mostly used in the hidden layers and the average of the outputs in each layer is close to zero. The mathematical expression [18]:

$$tangent = \frac{e^{2x}-1}{e^{2x}+1} \tag{6}$$

- **Rectified Linear Unit (ReLu)**

  It is a nonlinear function that can improve performance and reduces the number of computations during training by reducing the state of vanishing and error gradient problems. When large numbers of layers are used in the hidden, the ReLu can enhance training because of its speed [18]. It returns 0 if it receives any negative input, and returns the same value if it receives any positive value.

$$ReLu = \max(0, x) \tag{7}$$

- **Softmax**

  The Softmax function is also called the normalized exponential function. It normalizes the input into a probability distribution that sums to 1. It can be computed [18] as:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{8}$$

(c) Optimization function

Training the deep neural network requires changing the model parameters (weights, learning rate) until the minimal loss (difference between predicted output and the real output) is obtained and the most accurate result is achieved. This process of tuning is called the optimization process. The optimization functions are the algorithms or methods used during the optimization process. The commonly used optimizers are:

- **Stochastic Gradient Descent (SGD)**: This is a gradient-based optimizer extended from Gradient Descent. The optimizer takes randomly a small set of the training samples instead of the whole samples per iteration. SGD uses only the learning rate for parameter updating.
- **Root Mean Square Propagation (RMSProp) Optimisation:** This functions in a similar manner as the gradient descent with momentum but the gradient calculation is different. The momentum will allow the accumulations of the gradient of the past steps to determine the direction to move.
- **Adaptive Moment Estimation (Adam)**: This optimizer is used to minimizes loss in binary classification and multi classification. It has the fastest convergence when compared to other optimizer functions. This is because Adam uses three parameters namely a weighted average of gradient, a weighted average of squared gradient and a learning rate to update the parameters during training at every iteration.

Building the neural network requires choosing two important hyperparameters (batch size and number of epochs) for the optimization process. The two parameters contribute to the improvement of the model performance during training if properly fine-tuned.

(d) Regularization: Regularization is a method used to modify the deep learning algorithms. This involves tuning the hyper-parameters of a deep neural network that are used to control the learning process of a neural network. They are set before the start of the learning process e.g number of epochs, number of branches, dropout rate, etc. In deep learning, a commonly used regularization is the inclusion of a dropout layer to the layers of the network with a dropout rate of between zero to one. A *dropout layer* is a form of regularization that helps to reduce overfitting so that the model can generalize well on unseen data.

When it is applied, it means the training is performed on a small network compared to the original network, and since a smaller network can be less flexible, the possibility of

overfitting is reduced. The idea is that it randomly drops out a few neurons or sets the neurons to zero thereby reducing computation in the training process. Also, during the process of dropout, the number of weights updated during the training is reduced to allow other weights to participate in the learning process. This helps the weight values to spread out better at the final stage thereby reducing overfitting in the network.

(e) Iteration: This is the number of batches needed to complete one epoch. The number of iterations depends on the size of the batch, for a large batch the number of iterations will be small and does not need to change much before the neural network learns well.

(f) Batch size: This accounts for the training dataset present in a single batch**.** Since one epoch is too big to be fed into the computer at once, and because of system limitations, the dataset can be divided into several batches which determines the number of iterations.

(g) Epoch: One epoch is attained when an entire data is passed in and out of the neural network at once. Continuous feeding of the training data to the neural network can help to improve the weights. Updating of the parameter is done after each iteration. An increase in the number of epochs can generally increase accuracy and lower the loss.

(h) **Learning rate**: This parameter is important in the configuration of the deep neural network, and takes smaller values in the range between 0 and 1. It controls how fast the model adapts to the problem. A smaller learning rate would require more training epochs in relation to the smaller changes made to the weight on each update, and on the other hand, larger learning rates would require fewer training epochs.

2.7 Related works

There have been many analytics research works in intrusion detection systems using deep learning in the previous years. This area of research has started to gain popularity because of its ability to learning and development, which makes them very effective and efficient in tackling the alarming increase in the number of unpredictable attacks. Using deep learning techniques in the design of IDS gives it extra performance with high accuracy. There have been different approaches to intrusion detection using deep learning algorithms.

Salama et al [40] proposed a hybrid approach of IDS. Their works entail the use of Restricted Boltzmann Machine (RBM) as the feature reduction method, followed by a Support Vector Machine (SVM) as the classifier. The NSL-KDD dataset was used which has 22 training attack types and 17 types in the test data. The hybrid approach shows a higher percentage of classification better than Support Vector Machine as a stand-alone algorithm. Alom et al [41] in 2015 proposed the design of an effective intrusion detection system that is intelligent and can interpret intrusion from incoming network traffic. They trained a deep belief neural network (DBN) using the NSL-KDD dataset, the model was able to identify any kind of unknown attacks in a dataset that was fed into it, and had a detection accuracy of about 97.5%.

Alazab et al [42] conducted a malicious network traffic classification experiment using the UNSW-NB15 dataset in a convolutional neural network architecture. They first analysed+ the experiment with a fully connected neural network (3 layers of 256, 1024, and 7 nodes) as a benchmark for comparison to the CNN techniques. Though the fully connected neural network achieved the highest accuracy on the test data, the CNN performance accuracy was close as well with five to ten times fewer trainable parameters.

Kim et al [43] proposed an intrusion detection system model using Long Short-Term Memory (LSTM) architecture to a recurrent Neural Network (RNN). They trained the model using a KDD Cup 99 dataset (with 41 features) and trained the network with a time step size of 100, batch size 50, and epoch 500 and obtained 98.93% accuracy on the test data as the best performance. IDS on a wireless network based on deep learning using Feed forward deep neural network (FFDNN) was proposed by Kasongo et al [44]. This DNN was trained using the NSL-KDD dataset and its performance was compared to conventional machine learning algorithms (support vector machine, decision tree, k-nearest neighbor, and naïve Bayes). The FFDNN performance was best with an accuracy of 86.62% on test data. Naseer et al [45] developed an intrusion detection system model using three different DNNs (CNN, autoencoder, RNN) on the NSL-KDD dataset. As a baseline for comparison, the same experiment was conducted using three conventional machine learning algorithms (decision tree, support vector machine, and k-nearest neighbor). Their training performances were evaluated on two different test datasets (NSLKDDTest+ and NSLKDDTest21), and the overall result showed that LSTM had the best performance at 89% and DCNN was 85%. The decision tree, support vector machine, and k-nearest neighbor had a tie at 82%.

3 Design and Methods

3.1 Methods and Material

The research is based on an empirical approach that deals with the detection of intrusions in network activities using machine learning techniques.

3.1.1 Tools and Environment

The implementations of the Machine learning algorithms and the Feedforward Deep Neural network were performed using the Google Colaboratory software platform. It was selected because it is an open-source cloud-based jupyter notebook that is free and offers some awesome advantages that made it possible to train machine learning and deep learning models on CPUs, GPUs, and TPUs. These advantages include:

1. Provides free GPU/TPU
2. Zero configuration is required
3. Can access drive directly
4. Uploading files locally is easy
5. It comes with the following pre-installed libraries, NumPy; Pandas; Python; Matplotlib; Keras; Tensor flow, and Scikit learn which were necessary for the experiment.

But its limitation is that session restarts after 12 hours.

3.1.2 Dataset Description

The training and test data used for this research work was extracted from the NSL-KDD dataset. It was downloaded from the web page of the University of New Brunswick (Canadian Institute for Cybersecurity) [46]. The NSL-KDD dataset was proposed by Tavallae et al [47]. It is an updated version of the KDD Cup99 dataset and is recommended to solve some of the essential problems found in the KDD'99 dataset [47]. It is an effective standard that researchers can use to compare different types of intrusion detection systems methods [46] and also to design intrusion detection systems whether host-based or network-based.

Compared to the KDD'99 dataset, the NSL-KDD dataset have the following advantages:

1. Redundant records present in the train set has been removed to avoid the classifier from been biased
2. Duplicate records removed from the test set

3. The quantity of selected records from each difficulty level domain is inversely proportional to the level of records in the original KDD dataset.
4. There is a sufficient number of records available in the train and test dataset.

Table 4: List of NSL-KDD dataset files and their descriptions [48]

S/n	Name of the file	Description
1	KDDTrain+.ARFF	The full NSL-KDD train set with binary labels in ARFF format
2	KDDTrain+.TXT	The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3	KDDTrain+_20Percent.ARFF	A 20% subset of the KDDTrain+.arff file
4	KDDTrain+_20Percent.TXT	A 20% subset of the KDDTrain+.txt file
5	KDDTest+.ARFF	The full NSL-KDD test set with binary labels in ARFF format
6	KDDTest+.TXT	The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7	KDDTest-21. ARFF	A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
8	KDDTest-21.TXT	A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21

Each record contains 41 types of features that are assigned to attack or normal type. The features are subdivided into three (3) types of attribute value types (Nominal, binary and numeric).

Table 5: Basic Features of each Network Connection Vector [48][49]

No	Feature Name	Description
1	Duration	Duration of the connection
2	Protocol type	Connection protocol (e.g. TCP, UDP, ICMP)
3	Service	Destination service
4	Flag	Status flag of the connection
5	Src_bytes	Bytes sent from source to destination
6	Dst_bytes	Bytes sent from destination to source
7	Land	1 if a connection is from/to the same host/port; 0 otherwise
8	Wrong_fragment	Number of wrong fragments
9	Urgent	Number of urgent packets
10	Hot	Number of “hot” indicators
11	Num_failed_logins	Number of failed logins
12	Logged_in	1 if successfully logged in; 0 otherwise
13	Num_compromised	Number of “compromised” conditions
14	Root_shell	1 if root shell is obtained; 0 otherwise
15	Su_attempted	1 if “su root” command attempted; 0 otherwise
16	Num_root	Number of “root” accesses
17	Num_file_creations	Number of file creation operations
18	Num_shells	Number of shell prompts
19	Num_access_files	Number of operations on access control files
20	Num_outbound_cmds	Number of outbound commands in an FTP session
21	Is_host_login	1 if login belongs to the “hot” list; 0 otherwise

Table 6: Basic Features of each Network Connection Vector [48][49]

No	Feature Name	Description
22	Is_guest_login	1 if the login is the "guest" login; 0 otherwise
23	count	Number of connections to the same host as the current connection in the past 2 seconds
24	Srv_count	Number of connections to the same service as the current connection in the past two seconds
25	serror_rate	% of connections that have "SYN" errors
26	Srv_serror_rate	% of connections that have "SYN" errors
27	Rerror_rate	% of connections that have REJ errors
28	Srv_rerror_rate	% of connections that have REJ errors
29	Same_srv_rate	% of connections to the same service
30	Diff_srv_rate	% of connections to different services
31	Srv_diff_host_rate	% of connections to different hosts
32	Dst_host_count	Count of connections having the same destination host
33	Dst_host_srv_count	Count of connections having the same destination host and using the same service
34	Dst_host_same_srv_rate	% of connections having the same destination host and using the same service
35	Dst_host_diff_srv_rate	% of different services on the current host
36	Dst_host_same_src_port_rate	% of connections to the current host having the same src port
37	Dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts
38	Dst_host_serror_rate	% of connections to the current host that have an S0 error
39	Dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error
40	Dst_host_rerror_rate	% of connections to the current host that have an RST error
41	Dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error

The features are grouped into four categories [49]:

- **Basic features:** Features obtained from the packet header (TCP/IP connection) without the payload inspection. Features 1 to 9 is the basic features.
- **Content features:** They are generated from the payload of TCP segments and comprises features 10 to 22. The number of failed logs in an attempt is an example.
- **Time-based traffic features:** These features capture properties that mature within a window interval such as features 23 t0 31.

  It is divided into two sub-group

  a) "same host" features: this checks connections in the past 2 seconds that have the same destination host as the current connection, and then evaluates the statistics associated with the protocol behavior, service, etc.

  b) "same service" features: it checks only the connections in the past 2 seconds that have the same service as the current connection.
- **Host-based traffic features:** Contain features (32 to 41) that are designed to measure attack within intervals longer than 2 seconds.

The 42nd feature contains data which are part of the 5 different classes of network connection vectors. They divided into one normal class and four attacks class. The four attack classes are sub-grouped into DoS, R2L, Probe, and U2R [47] [48].

Table 7: Attribute value Type [48]

Type		
Features		
Nominal	**Binary**	**Numeric**
Protocol_type (2) Service (3) Flag (4)	Land (7), logged_in (12), root_shell (14) su_attempted (15) is_host_login (21) is_guest_login (22)	Duration (1) src_bytes (5) dst_bytes (6) wrong_fragment (8) urgent (9), hot (10) num_failed_logins (11) num_compromised (13) num_root (16) num_file_creations (17) num_shells (18) num_access_files (19) num_outbound_cmds (20) count (23) srv_count (24) serror_rate (25) srv_serror_rate (26) rerror_rate (27) srv_rerror_rate (28) same_srv_rate (29) diff_srv_rate (30) srv_diff_host_rate (31) dst_host_count (32) dst_host_srv_count (33) dst_host_same_srv_rate (34) dst_host_diff_srv_rate (35) dst_host_same_src_port_rate (36) dst_host_srv_diff_host_rate (37) dst_host_serror_rate (38) dst_host_srv_serror_rate (39) dst_host_rerror_rate (40) dst_host_srv_rerror_rate (41)

Table 8: Attack types of the different attack classes in NSL-KDD dataset [48]

Attack class	Attack types
DoS	Back, Land, Smurf, Neptune, Pod, Udpstorm,Teardrop, Apache2, worm.
Probe	Satan, Mscan, Ipsweep, Nmap, Saint, Portsweep.
R2L	Ftp_write, Snmpguess, Imap, Sendmail, Pht, Multihop, Warezmaster, Guess_password, warezclient, Spy, Xlock, Xsnoop, Httptunnel, Snmpget attack.
U2R	Rootkit, Loadmodule, Buffer_overflow, Perl, Xterm, sql attack.

Table 9: Details of Normal and Attack data in different types of NSL-KDD dataset [48]

Dataset type	Number of Records					
	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+20%	25192	13449 (53%)	9234 (37%)	2289 (9.16%)	11(0.04%)	209 (0.8%)
KDDTrain+	125973	67343 (53%)	45927 (37%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
KDDTest+	22544	9711 (43%)	7458 (33%)	2421 (11%)	200 (0.9%)	2654 (12.1%)

3.1.2.1 Data Pre-processing

Data pre-processing is an essential step that is needed before training the model. It helps to remove the outliers, missing values, duplicates, and converts categorical variables into numerical values, and normalizes the data. For this research, the KDDTrain+ dataset was used as the training set and KDDTest+ as the test dataset.

3.1.3 Feature Encoding

The features were encoded in three different methods. *Label encoding* and *One Hot encoding* techniques were used to convert the categorical features into numerical values in two different situations, that is building two similar models with different encoding technology. The *Binarization* technique was used to convert the attack class (target variable) into 0's and 1's to make the classifier algorithm more efficient.

(a) Binarization: The Normal situations were assigned values 1's and other attack types were assigned 0's irrespective of whether it is from DoS, Probe, U2R, or R2L.

(b) LabelEncoder: Label-Encoder is used to convert the three categorical features protocol_type, flag, and service, into numerical values. After label encoding, there was no increase in the number of features as the feature dimensions remain 41. The output of the LabelEncoder is still in the form of a data frame.

(c) One-Hot-Encoding is used to convert the three categorical features protocol_type, flag and service into numerical values in the form of binary. The protocol_type feature has three attributes: TCP, UDP, and ICMP. One-hot-encoding converts them into binary vectors of [1,0,0], [0,1,0], [0,0,1], respectively. In the same manner service and flag features. After One-hot-encoding, the KDDTrain+ datasets are mapped from 41-dimensional features to 122-dimensional features (38 continuous, and 84 binary values related to the categorical features). The output of the One-Hot-Encoding is a NumPy array.

3.1.4 Feature Scaling

The input was normalized so that they will be centred around zero and have the same scale. This is because the value of the input influences the update rule. [50] The min-max normalization method is used to scale the value $X_{i,j}$ between the range [0,1].

Let value after normalization = X'

Value before normalization = X

$$\frac{X'-0}{1-0} = \frac{X-min}{max-min} \tag{9}$$

$$X' = \frac{X-min}{max-min} \tag{10}$$

3.1.5 Feature Selection

Feature selection selects the most important features and discards features that are not relevant so that the model would be easier to understand, and as well run faster and perform well. Two different selection techniques were used, Random Forest and Boruta.

- **Random Forest**: RF is an embedded method that combines the qualities of the filter and wrapper methods. It has many (4 - 12) hundred decision trees, each tree is built over a random extraction of the observations from the dataset and the features. It divides the dataset

into two buckets, each bucket having observations that are closely related among themselves, but different from the ones in the other bucket. The measure of impurity is the Gini Impurity, and how pure each bucket determines the importance of each feature. This method was easy to interpret and can generalize well.

- **Boruta** is built around Random Forest and can be used on Random Forest model, XGBoost, and Regression models. Its working principle involves creating shadow features (random features and shuffle values in columns), A Random Forest model is trained on the data and feature importance is calculated via Mean decreasing Gini impurity. It checks if real features have higher importance compared to shadow features, if the original feature performs better, then it is marked as important. This is repeated for every iteration. This is a good method of feature selection because features do not compete among themselves but instead compete with a randomized version of themselves (shadow features), where the importance of each original feature is compared with a threshold (highest feature importance recorded among the shadow features). In Boruta, a feature is important if it does better than the best-randomized feature.

3.2 Classification methods

3.2.1 Binary Classification

The conventional machine learning algorithms and deep neural network algorithms are presented for binary classification of the KDDTrain+ dataset. Specifically, K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Naïve Bayes (NB), Logistic Regression (LR), and a Feedforward Deep Neural Network (FFDNN) are implemented. The classification was done for two different cases:

(i) Using all 41 features of the KDDTrain+ dataset to carry out a binary (Normal and Attack) classification and evaluating the performance of the models (ML and DL algorithms) on the KDDTest+ dataset.

(ii) Using reduced features obtained through feature selection to carry out a binary (Normal and Attack) classification and evaluating the performance of the FFDNNs models on the KDDTest+ dataset.

3.2.2 Multi-class classification

FFDNN multi-class classification of the intrusion detection on KDDTain+ was considered for future work, already mentioned that the attacks were divided into four categories, DoS, Probe, R2L, and U2R. The FFDNN model will be trained to classify the various anomalies into their different classes, and the prediction on the test data will confirm how well the model has achieved the desired accuracy. and the model was trained on them. However, based on the result of the binary classification, there is no doubt that the Feedforward deep learning model will perform well.

3.3 Statistical measures

The ground truth value is needed in the evaluation to estimate the different statistical measures. It is composed of a set of connection records labelled either Normal or Attack in the case of binary classification. The evaluation of the metrics is based on the four output values of the confusion matrix in Table 10 obtained by considering the calculated predicted class versus the actual class (ground truth).

Table 10: Confusion matrix

Actual class (Ground Truth)		**Predicted class**	
		Normal	**Attack**
	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

Considering the number of Normal and Attack connection records in the test dataset, the following terms are used for determining the quality of the classification models [7] [18]:

• **True Positive** (TP) - the number of connection records correctly classified to the Attack class.
• **True Negative** (TN) - the number of connection records correctly classified to the Normal class.
• **False Positive** (FP) - the number of Normal connection records wrongly classified to the Attack connection record.
• **False Negative** (FN) - the number of Attack connection records wrongly classified to the Normal connection record.

The following evaluation metrics as listed below are evaluated based on TP, TN, FP, and FN.

1) **Accuracy (acc)**: It estimates the ratio of the correctly recognized connection records to the entire dataset. The higher the accuracy, the better the machine learning model prediction is (Accuracy ∈ [0, 1]). It is a good measure for the test dataset that contains balanced classes and is defined as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

2) **Precision (P)**: Also called the positive predictive value (PPV). It calculates the ratio of the correctly identified attack connection records to the number of all identified attack connection records. Higher precision indicates that the machine learning model performed better (Precision ∈ [0, 1]). Precision is defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{12}$$

3) **F1-Score (F1)**: It is also the F1-Measure, it is defined as the harmonic mean of Precision (P) and the true positive rate (Recall). If the F1-Score is higher, the model is better (F1−Score ∈ [0, 1]). F1-Score is defined as follows:

$$F1 - Score = 2 \times \left(\frac{Precision \times Recall}{Precision+Recall}\right) \tag{13}$$

4) **True Positive Rate (TPR)**: It is also called Recall or Sensitivity, or Probability of Detection (P_D). It calculates the ratio of the correctly classified Attack connection records to the total number of Attack connection records. If the TPR is higher, the machine learning model is better (TPR ∈ [0, 1]). TPR is defined as follows:

$$TPR = \frac{TP}{TP+FN} \tag{14}$$

5) **False Positive Rate (FPR)**: It is also called False Alarm Rate (FAR or Fall-Out), It calculates the ratio of the Normal connection records flagged as Attacks to the total number of Normal connection records. Lower FPR indicates that the machine learning model is better (FPR ∈ [0, 1]). FPR is defined as follows:

$$FPR = \frac{FP}{FP+TN} \tag{15}$$

6) **Area under the Curve**: This is the size of the area under the Receiver Operating Characteristics (ROC) curve. ROC is a graph plotted between the TPR on the y axis to FPR on the x-axis across different thresholds. AUC is a good measure for evaluating the performance of the machine learning models. If the AUC is higher, the machine learning model is better.

$$AUC = \int_0^1 \frac{TP}{TP+FN} d \frac{FP}{TN+FP} \quad (16)$$

3.4 Model Implementation

The model implementation takes into consideration the results obtained on the evaluation of the test data (KDDTest+) set using statistical measures as mentioned in *Section 3.4*. The KDDTrain+ dataset was divided into two parts, namely a training dataset and a validation dataset. The test dataset has a different folder. The training dataset is used for training the model, and its prediction accuracy is measured on the validation set. Once I am satisfied with my selected model type and hyperparameters, my next step was to predict the test dataset using the new model

Table 11: Distribution of training and testing records

Dataset type		**Number of Records**					
		Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+		125973	67343 (53%)	45927 (37%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
KDDTest+		22544	9711 (43%)	7458 (33%)	2421 (11%)	200 (0.9%)	2654 (12.1%)

3.4.1 Implementation of Conventional ML models

KDDTrain+ dataset was trained using conventional machine learning algorithms, namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), and Logistic Regression (LR), their performances were evaluated on the KDDTest+ dataset and the classification reports and various performance accuracies are shown in *Table 13 & Table 14* were obtained from their confusion matrices respectively. After training the models using the training dataset, they were evaluated on the

validation datasets, after which predictions are made on the test dataset for final performance results from the different ML models.

3.4.2 Implementation of DNN models using Feed Forward Deep Neural Network (FFDNN)

The performance of the FFDNN was studied using the complete features (41 features) and the reduced features of the KDDTrain+ dataset. The KDDTest+ was used to predict training and validation accuracy. The validation loss and accuracy curves were used to monitor the behaviour of the model to avoid overfitting, with this principal criterion the trained model can either be accepted or rejected.

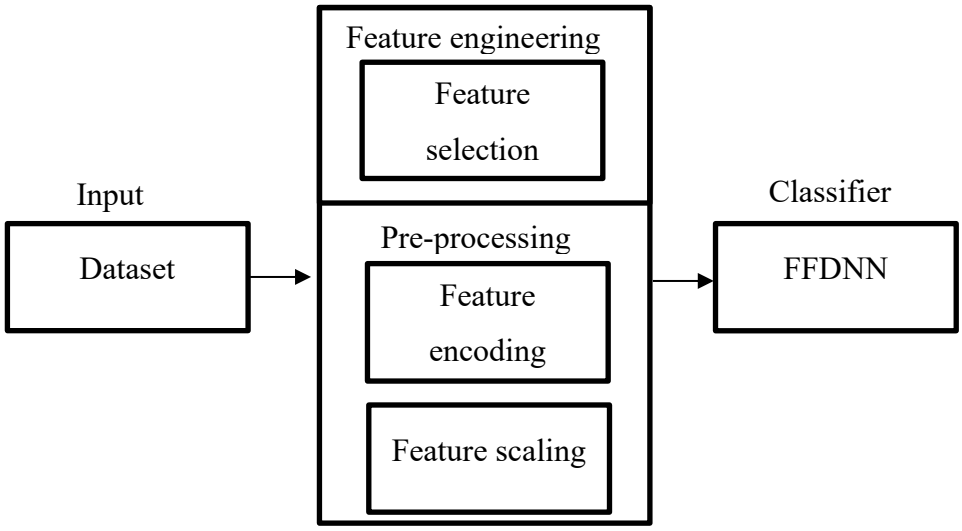

Figure 10: System flow of the FFDNN Classification model

The architecture of the FFDNNs was designed for binary classifications. They are made up of three layers of two hidden layers consisting of several neurons as listed in *Table 12*, and an output layer with one neuron in each model. Dropout layers were included to regularize the model during training to reduce overfitting. *ReLu activation function* was used in each hidden layer and the sigmoid activation function for binary classification was used in the output layer. The prediction loss function for sigmoid is defined using *binary cross-entropy*, estimated in [18] as

$$loss(pd, ed) = -\frac{1}{N}\sum_{i=1}^{N}[ed_i log pd_i + (1 - ed_i)\log(1 - pd_i)] \quad (17)$$

Where pd is a vector of predicted probability for all samples in the testing dataset, ed is a vector of an expected class label, values are either 0 or 1.
The FFDNN model is compiled with the *Adam optimizer* and "*accuracy*" as the metric for validation.

For the multi-class classification, the prediction loss was estimated using categorical cross-entropy given by in [18] as

$$loss(pd, ed) = -\sum_x pd(x)\log\left(ed(x)\right) \tag{18}$$

Where ed is a true probability distribution and pd is the predicted probability distribution.

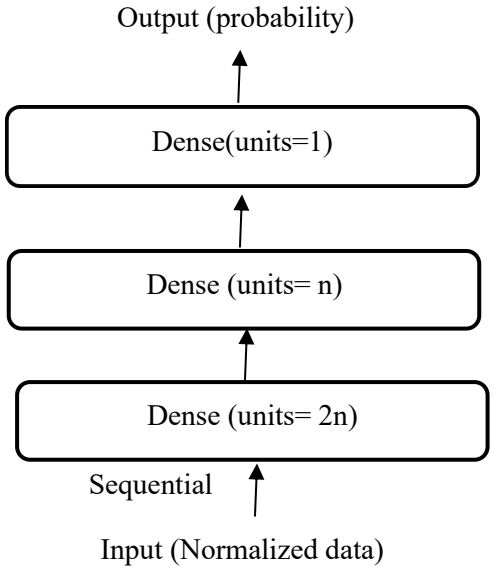

Figure 11: The General Architecture of FFDNNs as implemented in Keras

Table 12: Summary of Parameters in the different FFDNNs

Parameters	**FFDNN-1**	**FFDNN-2**	**FFDNN-3**
Number of features	41	13	122
Feature selection	No	Yes	No
Features Encoding	LabelEncode	LabelEncode	OneHot
Number of layers	3	3	3
Activation function in hidden layers	ReLU	ReLU	ReLU
Number of neurons in first hidden layers	64	32	512
Number of neurons in second hidden layers	32	16	256
Activation function in the output layer	Sigmoid	Sigmoid	Sigmoid
Dropout rate	0.2	0.2	0.2
Optimizer	Adam	Adam	Adam
Batch size	256	64	128
Number of epochs	16	20	20

4 Simulation and Results

4.1 Simulation Environment for Conventional Machine Learning algorithms

The simulation was carried out in the google colaboratory platform. All the necessary libraries (NumPy, Pandas, Scikit learn, Matplots, Seaborn) that are required to build the ML algorithms were imported. The first phase was data (train and test data) importation and pre-processing. There are no missing values and duplicates in the training and test data. The data preparation involves the binarization of the target labels into normal or attack (normal = 0 and attack = 1), encoding categorical data, passing data as NumPy array, and then normalizing the data and was passed into the ML algorithms.

Training data was divided into train and validation set to verify accuracy after fitting the model. Each ML model was trained with a 10 folds cross-validation, and the results were evaluated, after which prediction was made on the validation set to obtain accuracy, confusion matrix, and classification report. Obtaining satisfactory results, the test data was now passed into the models to obtain acceptable accuracies.

4.1.1 Performance Analysis of the Conventional ML algorithms on Test dataset

Table 13: Results obtained from Confusion Matrices for Conventional ML Algorithms

	DT (%)	**LR (%)**	**RF (%)**	**NB (%)**	**KN (%)**
Accuracy	81	76	77	77	77
True Negative (TN)	41	32	42	42	13
False Positive (FP)	2	11	1	1	30
False Negative (FN)	15	15	23	22	14
True Positive (TP)	42	42	34	35	43
True Positive Rate (TPR)	74	73	59	61	75
False Positive Rate (FPR)	5	26	3	2	70

Table 14: Precision, Recall, and F1 measure for binary classification on KDDTest+

Classifier	Precision (%)	Recall (%)	F1 Measure (%)	Accuracy (%)
Decision tree (DT)	95	74	83	81
Logistic regression (LR)	79	73	76	76
Random forest (RF)	96	59	73	77
Naïve Bayes (NB)	98	61	75	77
K-Nearest Neighbor (KN)	58	75	65	77

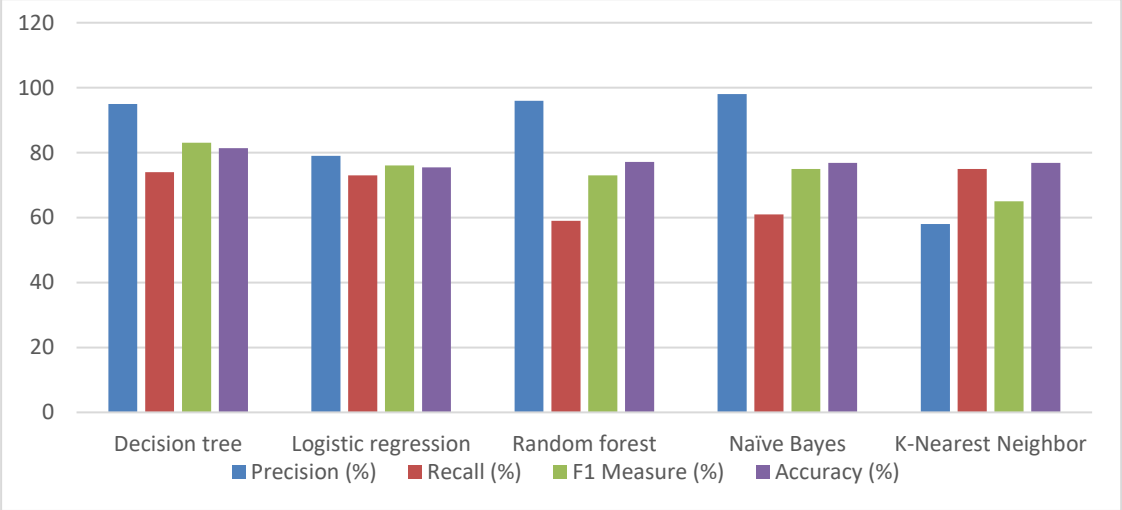

Figure 12: Precision, Recall, and F1 measure for binary classification on KDDTest+

4.2 Simulation Environment for FFDNN

Preparing the programming environment in the google colaboratory involves importing all the necessary libraries (NumPy, Pandas, Scikit learn, Keras, Tensor flow, Matplot, Seaborn) that are required to build the Deep Neural Network model. The first phase of the experiment was to import, clean, and prepare the training and test data. Based on the information about the data previously discussed in *section 3.1.2*, there are no missing values and duplicates. The data preparation involves the binarization of the target labels into normal or attack, encoding categorical data, passing data as NumPy array, and then normalizing the data to be ready to be fed into the deep neural network.

The Deep Neural Network was created using a sequential model from Keras API integrated into TensorFlow. The model architecture of three layers, the activation functions, number of neurons, and dropout layers of FFDNNs is shown in Appendix 1. Also available in appendix 2 is the model summary that shows the number of trainable parameters (weights and biases). Having defined the model architecture, the next step was to create the model compiler, and finally, the deep neural network was trained and its performance was evaluated both on the training dataset and validation dataset.

	loss	accuracy	val_loss	val_accuracy	epoch
11	0.026234	0.991119	0.022930	0.992618	11
12	0.024338	0.991933	0.022171	0.992300	12
13	0.023855	0.991625	0.021797	0.992141	13
14	0.022827	0.992300	0.021221	0.992657	14
15	0.020920	0.992905	0.020842	0.993134	15

	loss	accuracy	val_loss	val_accuracy	epoch
15	0.070985	0.976880	0.063872	0.977694	15
16	0.070069	0.976880	0.062467	0.978805	16
17	0.069766	0.977326	0.062520	0.978885	17
18	0.069520	0.977406	0.062085	0.978766	18
19	0.068753	0.977684	0.061201	0.978845	19

	loss	accuracy	val_loss	val_accuracy	epoch
15	0.008797	0.996491	0.007633	0.996825	15
16	0.008541	0.996674	0.008421	0.996785	16
17	0.008906	0.996634	0.008098	0.996706	17
18	0.008368	0.996602	0.007879	0.996706	18
19	0.008526	0.996864	0.008486	0.996745	19

Figure 13: Summary of the FFDNN-1, FFDNN-2 & FFDNN-3 results after each epoch

4.2.1 Performance Analysis of the FFDNNs on the training set

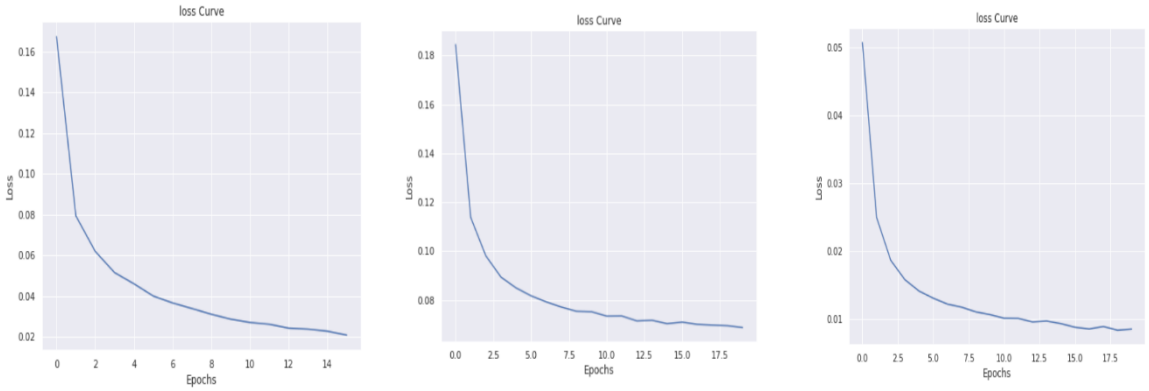

Figure 14: Loss curves for FFDNN-1, FFDNN-2 & FFDNN-3

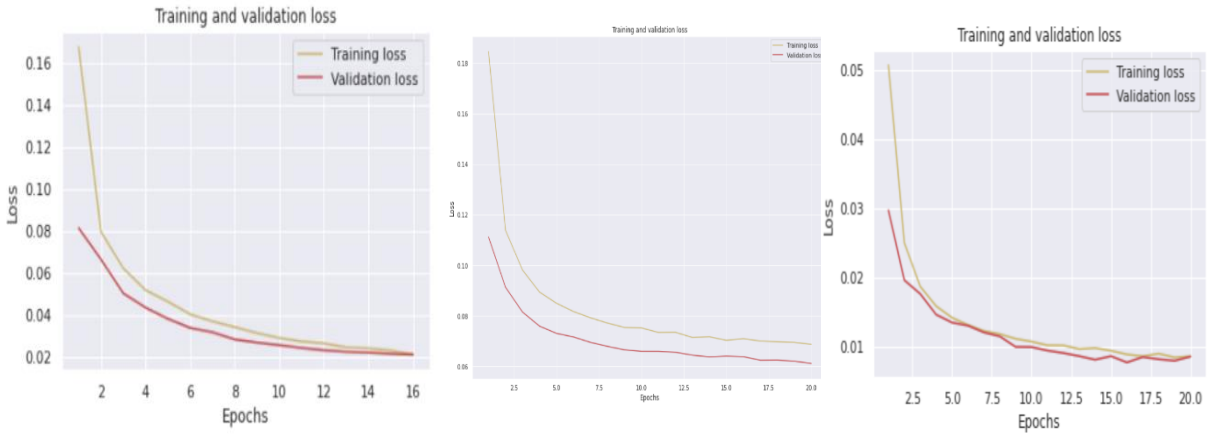

Figure 15: The training loss and validation loss rates during FFDNN-1, FFDNN-2 & FFDNN-3 training

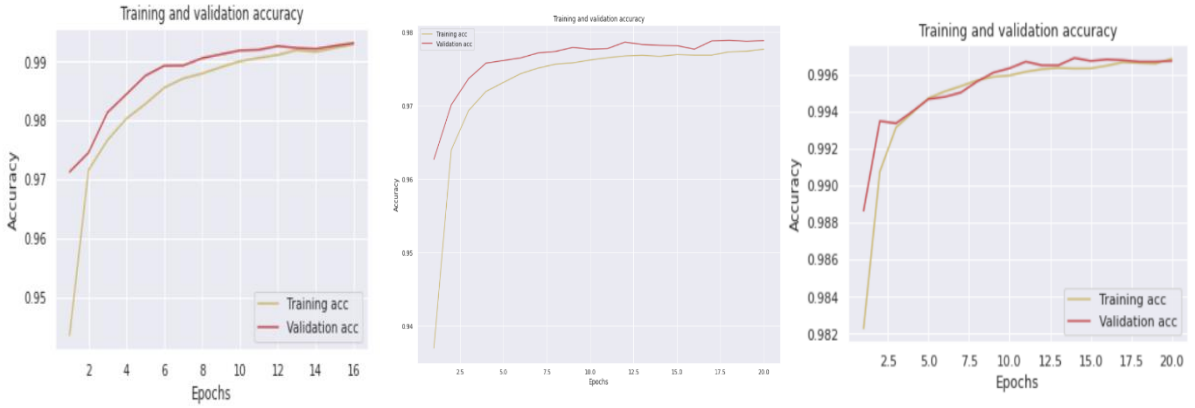

Figure 16: The training accuracy and validation accuracy curves for FFDNN-1, FFDNN-2 & FFDNN-3 models

4.2.2 Performance Analysis of the FFDNNs on the Test set

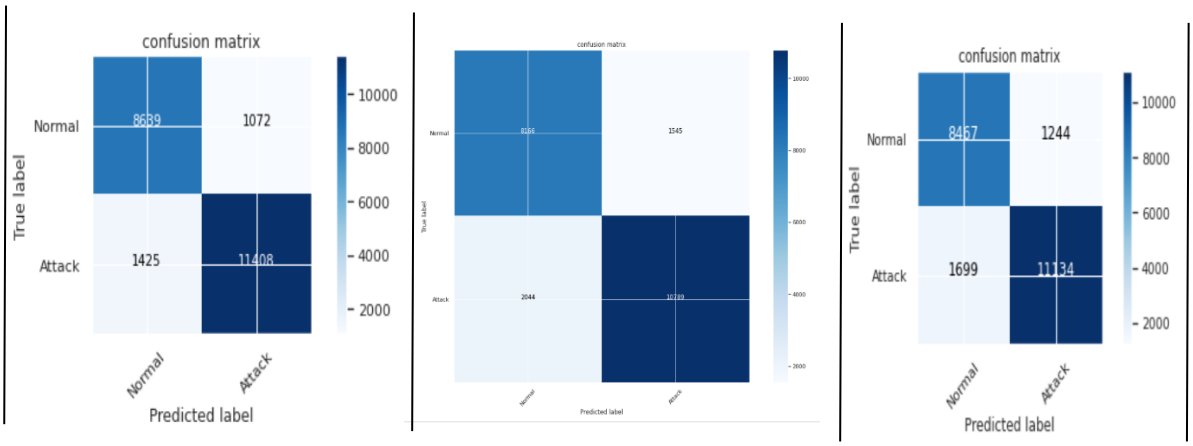

Figure 17: Confusion matrices yielded by FFDNN-1, FFDNN-2 & FFDNN-3

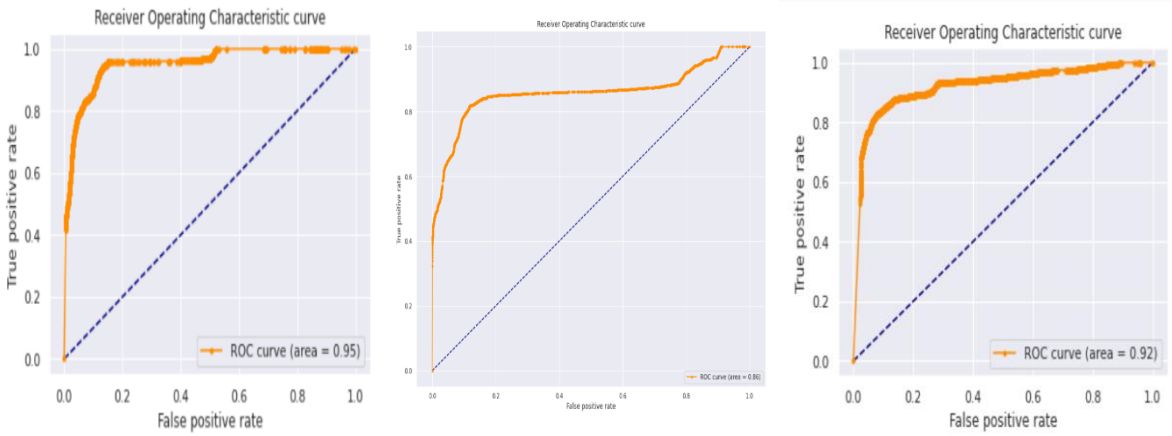

Figure 18: ROC curves show area under curve for FFDNN-1, FFDNN-2, FFDNN-3

Table 15: Results obtained from Confusion Matrices for the three FFDNNs

Metrics	**FFDNN-1**	**FFDNN-2**	**FFDNN-3**
True Negative (TN)	8639	8166	8467
False Positive (FP)	1072	1545	1244
False Negative (FN)	1425	2044	1699
True Positive (TP)	11408	10789	11134
Total	**22544**	**22544**	**22544**

Table 16: Metrics based on Confusion Matrix for FFDNNs in percentage

	FFDNN-1 (%)	**FFDNN-2 (%)**	**FFDNN-3 (%)**
Accuracy (Acc)	89	84	87
True Negative (TN)	38,3	36	37
False Positive (FP)	5	7	6
False Negative (FN)	6	9	8
True Positive (TP)	51	48	49
Misclassification (MC)	11	16	13
Area under Curve (AUC)	95	86	92
True Positive Rate (TPR)	89	84	87
False Positive Rate (FPR)	11	16	13

Table 17: Precision, recall, F1 measure and accuracy for FFDNN-1, FDNN-2, & FFDNN-3

DNN Classifier	**Precision (%)**	**Recall (%)**	**F1 Measure (%)**	**Accuracy (%)**
FFDNN-1	91	89	90	89
FFDNN-2	87	84	85	84
FFDNN-3	90	87	88	87

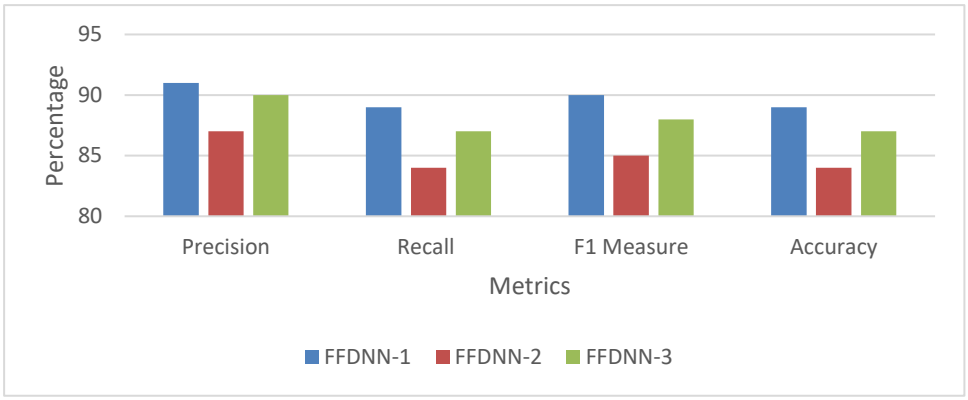

Figure 19: Graph on Precision, recall, F1 measure and accuracy for FFDNN-1, FFDNN-2, & FFDNN-3

Table 18: Performance Comparison of FFDNNs with five ML methods on KDDTest+

Classifier Methods	**Detection Accuracy (%)**
Decision tree	81
Logistic regression	76
Random forest	77
Naïve Bayes	77
K-Nearest Neighbor	77
FFDNN-1	89
FFDNN-2	84
FFDNN-3	87

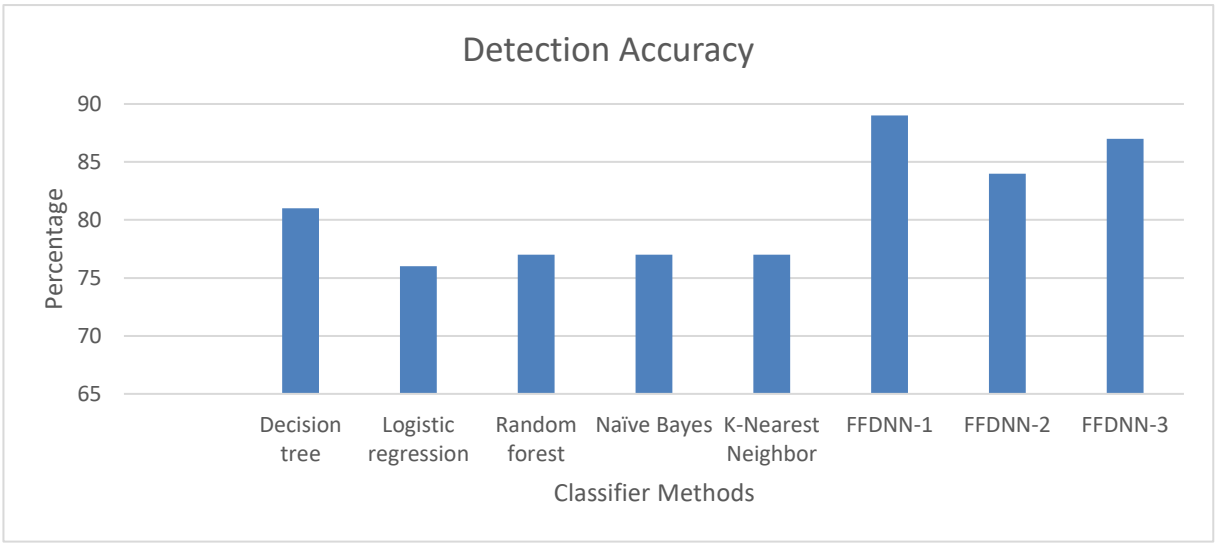

Figure 20: Graph on Detection Accuracy for Different ML Models

4.3 Results Analysis and Discussion

In this thesis, the researcher has conducted a comparative study of the machine learning approaches for intrusion detection using conventional machine learning algorithms and feedforward deep neural networks (FFDNN). My main idea was to develop an FFDNN as a classifier to classify network traffic intrusions in the NSL-KDD dataset as normal or attack while using the conventional machine learning approaches as my controlled experiment. The analysis was performed on five different ML algorithms (DT, LR, RF, NB, KNN) and their performances on the test data gave 81%, 76%, 77%, 77%, 77% respectively.

I also analysed three different FFDNNs, namely FFDNN-1, FFDNN-2, FFDNN-3 based on different hyperparameter values, feature reduction, and data encoding techniques. Their performances on the test data are 89%, 84%, and 87% respectively. Some important performance indicators, namely, accuracy, true positive rate (TPR = recall), false-positive rate (FPR), precision, F1 measure, and ROC area under the curve (AUC) are used as the bases to judge the model's performance.

Considering overall performances of all models, the FFDNN-1 performed best with the highest detection accuracy of 89% and precision of 91%, however, the precisions of the DT, RF, and NB were quite high even though their detection accuracy were not above 80% except DT that gave 81%. High precision indicates that the models are reliable compared to low precision that can result in a lot of false positives as in the case of the KNN classifier. Another point to note is that high recall does not necessarily mean the best performance, but rather higher F1-measure shows that the model had performed very well. This is because the F1 measure takes into account the harmonic mean of precision and recall, so it tells how accurate a model is. From the results shown in *Table 17* FFDNN-1 had an F1-measure of 90% which makes it the best model, other FFDNN variants also had a high F1-measure when compared to the F1-measure of the conventional machine learning algorithms.

Another important metric that shows that the FFDNNs performed well is their high ROC area under the curve (AUC), FFDNN-1, FFDDN-2, and FFDNN-3 in *Table 16* all had values of 95%, 86%, and 92% in that order. The higher the AUC value, the more optimal the model becomes (note that 100% indicates a perfect model). Also, it is good to know that the primary aim is to develop machine learning models that would maintain acceptable true positive rate

(TPR) or detection rate (DR) with low false alarm rate (FAR) or false positive rate (FPR), looking at *Table 13* and *Table 16*, you could see that DT, RF, and NB had very low FAR of 5%, 3% and 2% respective compared to FFDNNs 11%, 16%, and 13%. This is good even though their detection rates were not as high as the FFDNNs, however, there could still be an improvement in ensuring that the FFDNNs produce low FAR since they are scalable in terms of architecture, but this would involve more computational cost.

I would like to point out that any variant of the feedforward deep neural network (FFDNN-1, FFDNN-2, and FFDNN-3) has the capability as a classifier to classify network traffic intrusions with a high detection rate while offering a reduced FAR, it only depends on the model design, hyperparameters, and architecture.

Limitations in the ML approach for intrusion detection are high FAR and how to strike a balance between false positive and false negative in terms of intrusion detection policies or profiling. I believe this limitation can be solved by the development of a deeper neural network or the use of another variant of a deep neural network like the convolutional deep neural network (CNN).

4.4 Recommendations

The recommendations are based on my observations while performing my experiments.

(1) The hyperparameters (batch size, number of epochs, etc) are very important factors to consider with care when fine-tuning to avoid the deep neural network from overfitting or underfitting during training.

(2) Acceptable parameters can be achieved by carefully observing the shape of a plot of the training loss and validation loss rates, a plot of the training accuracy and validation accuracy rates, and also the plot of the loss curve function during training of the model.

(3) During prediction on the test dataset, the ROC curve can be used to visualize the binary classification of the DNN and to identify an ideal threshold that will yield optimal performance.

(4) The lower the batch size, the higher the training time and slower the training model, and vice versa.

(5) Improvement in the performance accuracy reduces the false positive (FP) & false negative (FN).

(6) Increasing the width of the neural network, i.e the higher the neurons in the hidden layers, the more parameter to be trained will increase, and this also improves the accuracy, TP, TN, and reduces FP, FN.

(7) Increasing the depth of the neural network, i.e increasing the number of hidden layers improves the statistical metrics.

5 Conclusion and Future work

In this thesis, Machine learning algorithms of binary classifiers applied to network intrusion detection have been discussed. As I mentioned earlier in section 1.3 as part of the thesis objectives, a deeper machine learning technique using the feedforward deep neural network (FFDNN) was developed. I first review the IDS concepts and the different IDS methods. They are followed by a discussion of some ML algorithms and their applications in network intrusion detection. Based on some important statistical metrics namely accuracy, true positive rate (TPR), false-positive rate (FPR), precision, F1 measure, and ROC area under the curve (AUC) the machine learning algorithms were evaluated on the test dataset and the FFDNN performed best with a performance accuracy of 89%. The conventional machine learning algorithms such as DT, RF, and NB also performed well but slightly lesser with performance accuracy of between 76% to 80%, however, had a low number of false alarms compared to FFDNN. The FFDNN is scalable in terms of its architecture (depth and width of the neural network) and size of the dataset which gives it an edge over the conventional machine learning, and therefore there is a possibility that false alarms can further be reduced, though this would lead to complexity in the model and high computational cost. During training, the training time in the FFDNN is reduced as compared to conventional machine learning.

The FFDNN can be deployed in real life to tackle the menace of malicious attacks on network activities. The experiments have demonstrated the effectiveness of the proposed technique in terms of correct classification and true positive rates of network records, so this confirms that the deep learning model would work efficiently to reduce the problem of false alarms. In the cybersecurity world, threats are constantly increasing as attackers have continued to devise new methods of attack daily. The deployment of the FFDNN in IDS will greatly help to tackle these attacks as it can detect unknown and novel threats. Finally, this proposed deep learning model is scalable which makes room for improvement to enhance its performance, so that as the change (emergent of new threats) keeps occurring the model will keep adapting to those changes.

In this thesis, the researcher conducted a binary classification using FFDNN on the NSL-KDD dataset, the same procedure can be applied for multi-class classification. The researcher hopes to continue his work in the future for the classification of the five different classes (Normal, DoS, Probe, U2R, R2L) by the FFDNN model.

The NSL-KDD dataset has been used for this thesis work; however, it is relatively outdated and may not cover many of today's network threats. So, I intend to experiment using alternative datasets in the future. Again, in today's digital world, the increase in the usage of IoT devices is constantly expanding the amount of data being generated on the network, it would not be worthwhile developing intrusion detection system using the conventional ML methods, and the increase in the complexity of the FFDNN to meet up this demand would consume computational resources. So, it is necessary to continue the research by using the convolutional deep neural network (CNN) and recurrent deep neural network (RNN) to build a deep learning model for an intrusion detection system that would be more efficient than using a feedforward deep neural network in terms of memory utilization, speed, and accuracy. The CNN deep neural network has the capability of automatic feature selection to select the best features from the dataset that would enhance the deep neural network performance, in like manner, the RNN is a time-variant deep neural network that will be very useful in automating an IDS to detect network intrusions in real-time.

References

[1] Macas, Mayra, and Chunming Wu. "Deep Learning Methods for Cybersecurity and Intrusion Detection Systems." *2020 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2020.

[2] Liu, Hongyu, and Bo Lang. "Machine learning and deep learning methods for intrusion detection systems: A survey." *applied sciences* 9.20 (2019): 4396. https://doi.org/10.3390/app9204396.

[3] Alghamdi, Mohammed I. "Survey on Applications of Deep Learning and Machine Learning Techniques for Cyber Security." *International Journal of Interactive Mobile Technologies* 14.16 (2020). https://doi.org/10.3991/ijim.v14i16.16953. [Accessed 25.04.2021].

[4] Microsoft Security Intelligence Report Special Edition 10 Year Review Key Findings Summary, https://www.studocu.com/fi/document/turun-yliopisto/system-application-and-security/tiivistelmat/microsoft-security-intelligence-report-special-edition-10-year-review-key-findings-summary/3091316/. [Accessed 02.04.2021].

[5] Vizitiu, Anamaria, et al. "Applying deep neural networks over homomorphic encrypted medical data." *Computational and mathematical methods in medicine* 2020 (2020). https://doi.org/10.1155/2020/3910250.

[6] Tariq, Muhammad Imran, et al. "A review of deep learning security and privacy defensive techniques." *Mobile Information Systems* 2020 (2020). https://doi.org/10.1155/2020/6535834.

[7] Berman, Daniel S., et al. "*A survey of deep learning methods for cybersecurity*." Information 10.4 (2019): 122. https://doi.org/10.3390/info10040122.

[8] Akbari Roumani, M., et al. "*Value analysis of cybersecurity based on attack types*." ITMSOC: Transactions on Innovation and Business Engineering 1 (2016): 34-39.

[9] Li, Jie, et al. "Machine learning algorithms for network intrusion detection." AI in Cybersecurity (2019): 151-179. https://doi.org/10.1007/978-3-319-98842-9_6 [Accessed 09.05.2021].

[10] Dua, Sumeet, and Xian Du. Data mining and machine learning in cybersecurity. CRC press, 2016., Introduction, Pg.1-114.

[11] Bhattacharyya, Dhruba Kumar, and Jugal Kumar Kalita. Network anomaly detection: A machine learning perspective. Chapman and Hall/CRC, 2019.

[12] Bioinformatics Web Development, "*Internet and Networks*", cellbiol.com [Online] Available: http://www.cellbiol.com/bioinformatics_web_development/chapter-1-internet-networks-andtcp-ip/the-tcpip-family-of-internet-protocols/ [Accessed:17 July. 2021].

[13] Li, Yadong, et al. "Research based on OSI model." *2011 IEEE 3rd International Conference on Communication Software and Networks*. IEEE, 2011. doi: 10.1109/ICCSN.2011.6014631.

[14] Wang, Jie, and Zachary A. Kissel. *Introduction to network security: theory and practice*. John Wiley & Sons, 2015.

[15] Depren, Ozgur, et al. "*An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks*." Expert systems with Applications 29.4 (2005): 713-722. https://doi.org/10.1016/j.eswa.2005.05.002. Accessed 25.07.2021.

[16] Weaver, Randy, Dawn Weaver, and Dean Farwood. Guide to network defense and countermeasures. Cengage Learning, 2013.

[17] Sarker, Iqbal H., et al. "*Cybersecurity data science: an overview from machine learning perspective*." Journal of Big data 7.1 (2020): 1-29. https://doi.org/10.1186/s40537-020-00318-5.

[18] Vinayakumar, Ravi, et al. "*Deep learning approach for intelligent intrusion detection system*." IEEE Access 7 (2019): 41525-41550. Doi: 10.1109/ACCESS.2019.2895334. [Accessed 25.04.2021].

[19] Thomas, Tony, Athira P. Vijayaraghavan, and Sabu Emmanuel. Machine learning approaches in cybersecurity analytics. Springer, 2020, https://doi.org/10.1007/978-981-15-1706-8.

[20] Alpaydin, Ethem. Introduction to machine learning. Third edition, MIT press, 2020.

[21] Xin, Yang, et al. "Machine learning and deep learning methods for cybersecurity." Ieee access 6 (2018): 35365-35381. Doi: 10.1109/ACCESS.2018.2836950.

[22] Ingre, Bhupendra, Anamika Yadav, and Atul Kumar Soni. "Decision tree-based intrusion detection system for NSL-KDD dataset." International conference on information and communication technology for intelligent systems. Springer, Cham, 2017.

[23] Azad, Chandrashekhar, and Vijay Kumar Jha. "Genetic algorithm to solve the problem of small disjunct in the decision tree-based intrusion detection system." International Journal of Computer Network and Information Security 7.8 (2015): 56-71.

[24] Kamarudin, Muhammad Hilmi, et al. "Packet header intrusion detection with binary logistic regression approach in detecting R2L and U2R attacks." 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec). IEEE, 2015. Doi: 10.1109/CyberSec.2015.28.

[25] Gao, Meijuan, Jingwen Tian, and Mingping Xia. "Intrusion detection method based on classify support vector machine." *2009 Second International Conference on Intelligent Computation Technology and Automation*. Vol. 2. IEEE, 2009. Doi: 10.1109/ICICTA.2009.330.

[26] Jiang, Jiaqi, et al. "A new intrusion detection system using class and sample weighted C-support vector machine." *2011 Third International Conference on Communications and Mobile Computing*. IEEE, 2011. Doi: 10.1109/CMC.2011.101.

[27] Atefi, Kayvan, Habibah Hashim, and Murizah Kassim. "Anomaly analysis for the classification purpose of intrusion detection system with K-nearest neighbors and deep neural network." *2019 IEEE 7th Conference on Systems, Process and Control (ICSPC)*. IEEE, 2019. Doi: 10.1109/ICSPC47137.2019.9068081.

[28] Almukaynizi, Mohammed, et al. "Patch before exploited: An approach to identify targeted software vulnerabilities." *AI in Cybersecurity*. Springer, Cham, 2019. 81-113. https://doi.org/10.1007/978-3-319-98842-9_4.

[29] Sharmila, B. S., and Rohini Nagapadma. "Intrusion Detection System using Naive Bayes algorithm." *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2019. Doi: 10.1109/WIECON-ECE48653.2019.9019921.

[30] Zhang, Jiong, Mohammad Zulkernine, and Anwar Haque. "Random-forests-based network intrusion detection systems." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.5 (2008): 649-659. Doi: 10.1109/TSMCC.2008.923876.

[31] Guowei, Z. H. U., et al. "Research on network intrusion detection method of power system based on random forest algorithm." *2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. IEEE, 2021. Doi: 10.1109/ICMTMA52658.2021.00087.

[32] Hagan, Martin T., Howard B. Demuth, and Mark Beale. *Neural network design*. PWS Publishing Co., 1997. (2nd Edition). [Online]. Available http://www.hagan.Okstate.edu/NNDesign.pdf . [Accessed 26.06.2021]

[33] Sani, Yusuf, et al. "An overview of neural networks use in anomaly intrusion detection systems." *2009 IEEE Student Conference on Research and Development (SCOReD)*. IEEE, 2009. Doi: 10.1109/SCORED.2009.5443289.

[34] Alla, Sridhar, and Suman Kalyan Adari. *Beginning anomaly detection using Python-based deep learning*. New Jersey: Apress, 2019.

[35] Geetha, R., and T. Thilagam. "A review on the effectiveness of machine learning and deep learning algorithms for cyber security." *Archives of Computational Methods in Engineering* 28.4 (2021): 2861-2879. https://doi.org/10.1007/s11831-020-09478-2.

[36] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

[37] Ravichandiran, Sudharsan. *Hands-on deep learning algorithms with python: master deep learning algorithms with extensive math by implementing them using tensorflow*. Packt Publishing Ltd, 2019.

[38] Bhagwat, Ritesh, Mahla Abdolahnejad, and Matthew Moocarme. *Applied deep learning with keras: Solve complex real-life problems with the simplicity of keras*. Packt Publishing Ltd, 2019.

[39] Galea, Alex, and Luis Capelo. *Applied Deep Learning with Python: Use scikit-learn, TensorFlow, and Keras to create intelligent systems and machine learning solutions*. Packt Publishing Ltd, 2018.

[40] Salama, Mostafa A., et al. "Hybrid intelligent intrusion detection scheme." *Soft computing in industrial applications*. Springer, Berlin, Heidelberg, 2011. 293-303.

[41] Alom, Md Zahangir, VenkataRamesh Bontupalli, and Tarek M. Taha. "Intrusion detection using deep belief networks." *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE, 2015. Doi: 10.1109/NAECON.2015.7443094. [Access 27.04.2021].

[42] Alazab, Mamoun, and MingJian Tang, eds. *Deep learning applications for cyber security*. Springer, 2019. https://doi.org/10.1007/978-3-030-13057-2_5. [Access 28.04.2021].

[43] Kim, Jihyun, et al. "Long short-term memory recurrent neural network classifier for intrusion detection." *2016 International Conference on Platform Technology and Service* (PlatCon). IEEE, 2016. Doi: 10.1109/PlatCon.2016.7456805. [Access 28.04.2021].

[44] Kasongo, Sydney Mambwe, and Yanxia Sun. "*A deep learning method with filter-based feature engineering for wireless intrusion detection system*." IEEE Access 7 (2019): 38597-38607. Doi: 10.1109/ACCESS.2019.2905633. [Access 28.04.2021].

[45] Naseer, Sheraz, et al. "Enhanced network anomaly detection based on deep neural networks." *IEEE access* 6 (2018): 48231-48246. Doi: 10.1109/ACCESS.2018.2863036. [Accessed 28.04.2021].

[46] University of New Brunswick, "Canadian Institute for Cybersecurity" NSL-KDD dataset [Online] Available: https://www.unb.ca/cic/datasets/nsl.html. [Accessed 23.04.2021].

[47] Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009. Doi: 10.1109/CISDA.2009.5356528.

[48] Dhanabal, L., and S. P. Shantharajah. "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms." *International journal of advanced research in computer and communication engineering* 4.6 (2015): 446-452. DOI: 10.17148/IJARCCE.2015.4696. [Accessed 23.04.2021].

[49] Jeya, P. Gifty, M. Ravichandran, and C. S. Ravichandran. "Efficient classifier for R2L and U2R attacks." International Journal of Computer Applications 45.21 (2012): 28-32.

[50] Ali, Peshawa Jamal Muhammad, et al. "Data normalization and standardization: a technical report." Mach Learn Tech Rep 1.1 (2014): 1-6. DOI: 10.13140/RG.2.2.28948.04489. [Accessed 13.05.2021].

Appendices

Appendix 1: FFDNNs Model in Keras Sequential API

```
model = Sequential()
model.add(Dense(64, input_dim = 41, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

FFDNN-1 Model

```
model = Sequential()
model.add(Dense(32, input_dim = 13, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

FFDNN-2 Model

```
model = Sequential()
model.add(Dense(512, input_dim = 122, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

FFDNN-3 Model

Appendix 2: Summary of FFDNNs Trainable Parameters

The higher the trainable parameters (weights and biases), the more complex the deep neural network

FFDNN-1

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_57 (Dense)             (None, 64)                2688
_________________________________________________________________
dropout_38 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_58 (Dense)             (None, 32)                2080
_________________________________________________________________
activation_38 (Activation)   (None, 32)                0
_________________________________________________________________
dropout_39 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_59 (Dense)             (None, 1)                 33
_________________________________________________________________
activation_39 (Activation)   (None, 1)                 0
=================================================================
Total params: 4,801
Trainable params: 4,801
Non-trainable params: 0
_________________________________________________________________
```

FFDNN-2

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                448
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
activation (Activation)      (None, 16)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 993
Trainable params: 993
Non-trainable params: 0
_________________________________________________________________
```

FFDNN-3

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 512)               62976
_________________________________________________________________
dropout_17 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_28 (Dense)             (None, 256)               131328
_________________________________________________________________
activation_18 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 257
_________________________________________________________________
activation_19 (Activation)   (None, 1)                 0
=================================================================
Total params: 194,561
Trainable params: 194,561
Non-trainable params: 0
_________________________________________________________________
```