ANOMALY- BASED INTRUSION DETECTION USING GENERATIVE ADVERSERIAL NETWORKS

A Thesis Presented

By

KALEAB AYELE FKADIE

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements for the Degree of Master of Science

in

Computer Science

February, 2021

ACCEPTANCE

ANOMALY- BASED INTRUSION DETECTION USING GENERATIVE ADVERSERIAL NETWORKS

By

KALEAB AYELE FKADIE

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

_____

Internal Examiner

_____

External Examiner

_____

Dean, Faculty of Informatics

February 2021

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Kaleab Ayele Fkadie
Name of Student

_____
Signature

Addis Ababa

Ethiopia

This thesis has been submitted for examination with my approval as advisor.

Asrat Mulatu (PhD)
Advisor

_____
Signature

Addis Ababa

Ethiopia

February 2021

Acknowledgments

First and foremost, praises and thanks to God, for his countless blessings throughout my life and this research. It has been a challenging journey and I would like to express my deep and sincere gratitude to my research advisor, Dr. Asrat Mulatu for his valuable time, guidance and comments which enabled me to gain good research experience.

I would also like to thank Dr.Michael Melese who gave me additional information on data processing and hyperparameter optimization for my research. This whole research would not be successful without his support.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. They have been my source of strength next to God. Last but not least, I would like to thank my friends and colleagues.

Table of Contents

List of Acronyms

AIDS	Anomaly Intrusion Detection System
ANIDS	Anomaly Network Intrusion Detection System
AUC	System Area under ROC curve
CPU	Central Processing Unit
DNN	Deep Neural Network
DOS	Denial of Service
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
HIDS	Host-Based Intrusion Detection System
IDS	Intrusion Detection System
JSD	Jensen-Shannon divergence
KDD	Knowledge Discovery Data
ML	Machine learning
NIDS	Network-Based Intrusion Detection System
NSL-KDD	Network Security Learning Knowledge Discovery Dataset
PCA	Principal Component Analysis
R2L	Remote to Local
RAM	Random Access Memory
RBM	Restricted Boltzmann Machine
ROC	Receiver Operating Characteristics
SFPR	System False Positive Rate
SIDS	Signature Intrusion Detection System
SM	Soft-Max Regression
STL	Self-taught Learning
TPR	True Positive Rate
U2R	User to Root
WGAN	Wasserstein Generative Adversarial Network

List of Figures

List of Tables

Abstract

Intrusion detection system (IDS) has become vital role in the field of IT Security due to cyber security safety in all human and machine pass through day to day activities**.** Intrusion detection methods based on the signature-based techniques have been used widely with limitation of identify new emerging threats. However, the progress of technology and the shortcomings of the intrusion detection system are influenced to upgrade IDS based on signature. Anomaly-based IDS are to establish a normal behavior profile and then define abnormal behaviors by their degree of abnormality from the normal profile. One of the techniques is used algorithms that support Deep Learning. Generative Adversarial Networks (GANs) have been widely studied and applied in anomaly detection within 6 years from first introduced in 2014 due to their advanced advantage in generating and learning higher-dimensional data which is had high number of features such as images, sounds and text. On this paper we had use current existing GAN and WGAN one of GAN variants for anomaly intrusion detection using NSL KDD dataset. On the training phase we have used pre-processed data fed to algorithms to train with default parameters that the classification model is build. On the validation phase we have considered of loss and accuracy of each batch of data training through with optimal parameters that gather from grid search over cross validation. Finally, the selected trained model is used to predict the test dataset. The evaluation result showed that the accuracy in classifying normal and attack. The results had shown on WGAN with accuracy of 89% prediction with default parameter and high prediction that performing with accuracy of 95.7% with optimized parameter.

Keywords: Deep Learning, Intrusion Detection System, Anomaly Detection, Neural Network, NSL KDD Dataset, Generative Adversarial Networks, Wasserstein Generative Adversarial Networks.

CHAPTER ONE

INTRODUCTION

1.1. Background of the study

Information Security is a key concern in the modern information process due to expanding computer technology with the threat it faces – loss of stored, processes and transmit information through the network. In the 90's, the beginning of an Internet era is providing a huge transformation on information technology, because of the data transmission and communication channel to become more easily usable [1]. It was a fixed network of computers that allowed the first millions of Internet users to communicate via e-mail. However, with the arrival of the Internet, personal computers and computer networks vulnerability increases to various kinds of attacks.

Heavy reliance on the Internet and worldwide connectivity has greatly increased the potential damage that can be inflicted by remote attacks launched over the Internet. And results of using Internet become with threat on information hijack and lose stored data. Intruders make use of the security breaches present in the system or network to attack it [2]. Intrusion is a purposefully illegal attempt to access information, manipulate information or render a system untrustworthy or inoperative.

Computer and network security is become a major concern in our daily life experience on the Internet. According to Kaspersky 2019 statistical reporting period, network attacks continued to be one of the most common types of attacks [3]. Kaspersky solutions repelled 975,491,360 attacks launched from online resources located all over the world. So, there should be mitigation for this threat. One of the major goals of network security is to detect an attack on network traffic. There are different ways to prevent and protect organizations network resources due to confidentiality, availability and integrity. Some of them are installing anti-virus software, firewalls, cryptography, intrusion detection system, and authentication and authorization. Among them, intrusion detection system (IDS) has been considered to be one of the most promising methods for defending complex and dynamic intrusion behaviors.

An intrusion detection system (IDS) is an active process or device that analyzes system and network activity for unauthorized and unauthenticated activity [4]. This is typically accomplished by automatically collecting information from a variety of systems and network sources, and then analyzing the information for possible security problems. There is no 100% guarantee to protect any data on the network which connected to Internet. Rather it is recommended to use different ways as an optional to mitigate any threat according to IEEE x.805 eight security dimensions map to the security threats. As stated on the paper [5], the eight security dimensions are access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability and privacy. In this paper, we have used to mitigate through communication security and access control based on the network traffic transaction records with in network traffic dataset.

Intrusion detection systems are identified the attack in to two ways according to detecting method which are Signature-based and Anomaly-based. Signature-based detection [4] process matches the signatures of samples using a signature database. The main problem in constructing signature detection systems is to design efficient signatures. Anomaly-based IDS are to establish a normal behavior profile and then define abnormal behaviors by their degree of deviation from the normal profile [6]. It has been used mostly to detect unknown attacks.

Nowadays most researchers in the area of network security focus to work on anomaly detection. Machine learning algorithms have been widely used in the improvement on IDS due to their high efficiency, flexibility and deploy ability. Currently, IDS based on machine learning techniques have become the mainstream. Rather than building a large signature database and the world become a data driven in every sector of government and private industries.

However, due to the huge quantification and complexity of malicious attacks, some shortcomings of traditional machine learning algorithms have been improved, such as the emphasis on processing low-dimensional data and the lack of response to high-dimensional data, and the dependence on manual features selection

Deep learning is branch of machine learning which highly powerful and effective in development of Intrusion Detection. The manual feature selection process simplify by deep learning and it has become a practical solution and implementation for machine learning tasks

that process a high-dimensional data which has high number of attributes can exceed the number of observations. In recent years, the application of deep learning algorithms in the field of intrusion detection has developed rapidly. Since Generative Adversarial Networks published paper in 2014 [7], GAN have shown their advanced advantage in generating higher-dimensional data such as images, sounds and text. GAN is mostly used in the field of Digital Image Processing and Computer Vision. However, it also can be used in intrusion detection. GAN is a combination of two neural networks which are Generator and Discriminator. A Generator alters malicious version of the input it was originally profile given and sends it to be classified by the Discriminator. The objective of the Generator is to bypass the IDS, and the objective of the Discriminator is to imitate the IDS on classifying inputs (normal or attack) and be responsible for response to the Generator. With this combination, both models are competing to win each other through adversarial. The attack is identified on the Discriminator Model.

In this research, to overcome the problems in identify unknown attacks using a set of selected deep learning algorithms are evaluated on NSL-KDD data set. Their performance are measured based on their detection rate and evaluation metrics. There are four major attack categories found on NSL-KDD datasets: Probe (information gathering), DoS (denial of service), and U2R (user to root) and R2L (remote to local). These four attacks have distinct unique execution dynamics and signatures. However, these four major attacks grouped into one attack category to identify from normal network traffic which is one of the drives for this research to investigate if certain detection algorithms are likely to demonstrate superior performance for a given network traffic records.

1.2. Motivation for Study

There are many intrusion detection techniques that are proposed by different researchers, some of which are reviewed in the literature review section like signature-based detection, anomaly-based host based, and network-based. Most Intrusion detection systems are designed and developed on signature-based approach which examines only known attacks. The detection process matches the signatures of samples using a signature database. It did not handle unknown attacks and weakness of currently available network security tools with regard to detecting intrusion. So it has to find a solution for continuous updating of the information to get all known

and unknown attacks in a network. The initiation takes place from information security issue through developing a secure way on an organization network through continuous study on the data from threats previously occurred and try to learn a machine to protect itself from new type threats. Thus, on this study a network-based approach with unidentified threats to determine and detect intrusions using a deep learning technique to learn feature from previously attacks and minimize the unknown attacks on the network.

1.3. Statement of the Problem

Intrusion Detection is one way of network monitoring mechanism to prevent the resources before further damage occurs [4]. IDS are design mostly on signature-based for known attacks, also there are depend on anomaly-based IDS for new threats. Detecting attacks masked by evasion techniques is a challenge for both Signature IDS and Anomaly IDS [8]. These techniques are malicious activities to avoid the detection of IDS. The ability of evasion techniques would be determined by the ability of IDS to bring back the original signature of the attacks or create new signatures to cover the modification of the attacks.

Robustness of IDS to various evasion techniques still needs further investigation. According to [4] improvements in machine learning algorithms are the main means to enhance the detection effect using different feature selection methods are intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.

Deep learning models directly learn feature representations from the original data, such as images and texts, without requiring manual feature engineering [9]. Thus, deep learning methods can execute in an end-to-end manner. For large datasets, deep learning methods have a significant advantage over shallow models. As in paper [9] stated that the ANIDS are developed as classifiers to differentiate the normal traffic from the anomalous traffic.

One of the activities is accompanied feature selection task is to extract a subset of relevant features from the traffic dataset to enhance classification results [8]. Feature selection helps in the elimination of the possibility of incorrect training through the removal of redundant features and noises.

To improve the strength of IDSs, adversarial learning becomes a novel approach [4]. Adversarial Learning can be used for attacks against IDS. Meanwhile, it is also a novel way to improve detection accuracy of IDS. The benefits of AIDS are strong generalizability and the ability to recognize unknown attacks. However it is vulnerable for high false alarm rate and an inability to provide possible reasons for an anomaly. So on this study by using generative adversarial learning can be comparison the effect on the detection efficiency based on the balanced and optimized features distributed on dataset NSL-KDD and measure the effectiveness of Generative Adversarial Networks (GAN) and its variance Wasserstein Generative Adversarial Networks (WGAN) using the IDS metrics to show the feasibility and comparison results [10].

There are different ANIDS challenges raised and solved in different studies. In this study, we will response the following:

1. Which factors are influence the performance of GAN and WGAN on ANIDS?
2. How can we increase the detection rate of Anomaly-based Intrusion Detection systems to detect unknown attacks?
3. Compare the efficiency of GAN and its variance WGAN on ANIDS Model.

1.4. Objectives

1.4.1.General Objective

The main objective of the study will be to design and implement a model for classification based on the Anomaly Network Intrusion Detection for network attacks.

1.4.2.Specific Objectives

The specific objectives of the study will be:

- Conduct a detail literature review to understand for deep learning in anomaly intrusion detection .
- To study different types of intrusion detection approaches.
- To design Anomaly-based Network Intrusion Detection using the selected algorithms.
- To conduct experiments to test and evaluate the performance of the model
- To perform an analysis on the optimal classification model based on the evaluation results.
- To identify factors that has influence on the models.

1.5. Scope

In this thesis work, conducted to design and implement a model for classification of network traffic into normal and attack based on the anomaly approach using deep learning techniques. It focuses on identifying possible network attacks, measuring the model efficiency and classifies the anomaly and normal profiles, not to prevent them. The study is designed to model and increase detection rate with examine unknown attacks. However, in this study host based intrusion detection is not included.

One of the limitations of this research work is that the dataset is used from Canadian Institute for Cyber security organization, and cannot directly implement the trained model to specific organization network. Which is due to the network infrastructure and configuration of one organization is different from the others. Also, the procedure for grid search with cross validation is needed long computational time.

1.6. Organization of the Thesis

The following is an overview of the structure of this thesis. First chapter gives an introduction to this research giving statement of the problem, thesis objectives, motivation and the scope of this work. This is followed by second chapter introduces the conceptual information on intrusion detection and related works in the field of anomaly intrusion detection using different detection techniques. It also discusses how intrusion detection systems are classified. The third chapter introduces the research methods, algorithms and dataset to use in this paper. The fourth chapter introduces the research experiment and evaluation which explores the study done including evaluation setup, criteria and performance analysis for both of the selected algorithms. And the fifth chapter introduces the discussion and concluding remarks on our study, present ideas for improvements and recommendations for future research are forwarded.

CHAPTER TWO

LITRATURE REVIEW

This Chapter is mainly concern on review literatures and basic concepts on IDS which are highly focus on anomaly detection using deep machine learning techniques and Generative Adversarial Network (GAN).

2.1. Overview of Intrusion Detection System

Network attacks are defined as a set of malicious activities to disrupt, deny, degrade or destroy information and service resident in computer networks [11]. A network attack is executed through the data stream on networks and aims to compromise the integrity, confidentiality or availability of computer network systems. Examples of computer attacks include viruses attached to emails, probing of a system to collect information, Internet worms, unauthorized usage of a system, and denial of-service by abusing a feature of a system, or exploiting a bug in software to modify system data [2].

Many attack recognition systems have been developed and are in use widely which inspect network data for any variation from the ordinary action of a system or user of the system [2]. Hackers have developed several mechanisms ranging from simple to sophisticated techniques to perpetuating their criminal acts.

In addition, the majority of attack, leverage on the loopholes found in some of the hardware and software components of the interconnected network systems [12]. Some might also look for an already recognized behavior of an attack within the data. These systems are termed as Intrusion Detection Systems (IDS) and use different techniques varying from statistical methods to machine learning algorithms. IDS are an important tool for network system to detect security holes in the network. Before further investigation, we will define important and usually used terms related with IDS from authors in [13], [14].

Network Intrusion refers to any unauthorized activity on a digital network. Network intrusions often involve stealing valuable network resources and almost always jeopardize the security of networks and their data.

Intruder: it can be any person, system or program that tries to or is successful to break into the network and perform illegal actions. The intruders may be an entity from outside or may be an inside user of the system trying to access unauthorized information.

Intrusion Detection: is the process of identifying and (possibly) responding to malicious activities targeted at computing and network resources by the observation of the information available about the state of the system and monitoring the user activities. Detection of break-ins or attempts by intruders to gain unauthorized access of the system is intrusion detection.

Anomaly intrusion detection (AID): is to determine if an activity is unusual enough to suspect an intrusion. A basic assumption of anomaly detection is that attacks differ from normal behavior. A normal behavior is the one used in the network which has valid access. Machine learning is used to adapt the environment however the one that tries to access outside from the normal or allowed is considered as malicious without changing the environment.

Intrusion detection system (IDS) is a kind of security management system for computers systems and networks. An Intrusion Detection System gathers the information from certain areas within a network or computers and analyzes it to find potential security breaches.

There are two types of IDS classification methods [4]: detection-based method and data source-based methods. Depending on how the intrusion is detected, there are two different types of IDS: signature-based (misuse) IDS (SIDS) and anomaly detection based IDS (ADIDS). SIDS [8]is based on pattern matching techniques to find a known attack; these are also known as Knowledge-based Detection or Misuse Detection.

In SIDS [8], matching methods are used to find a previous intrusion. In other words, when an intrusion signature matches with the signature of a previous intrusion that already exists in the signature database, an alarm signal is triggered. For SIDS, host's logs are inspected to find sequences of commands or actions which have previously been identified as malware. The main problem in constructing misuse detection systems is to design efficient signatures. The advantages of misuse detection are that it has a low false alarm rate and it reports attack types as well as possible reasons in detail; the disadvantages are that it has a high missed alarm rate, lacks the ability to detect unknown attacks, and requires maintaining a huge signature database.

Figure 2.1 shows how a typical misuse or signature detection system works [15]. These detection systems execute algorithms that attempt to match learned patterns or signatures from past attacks with the current activities in a network in order to detect any possible attack or malicious activities.

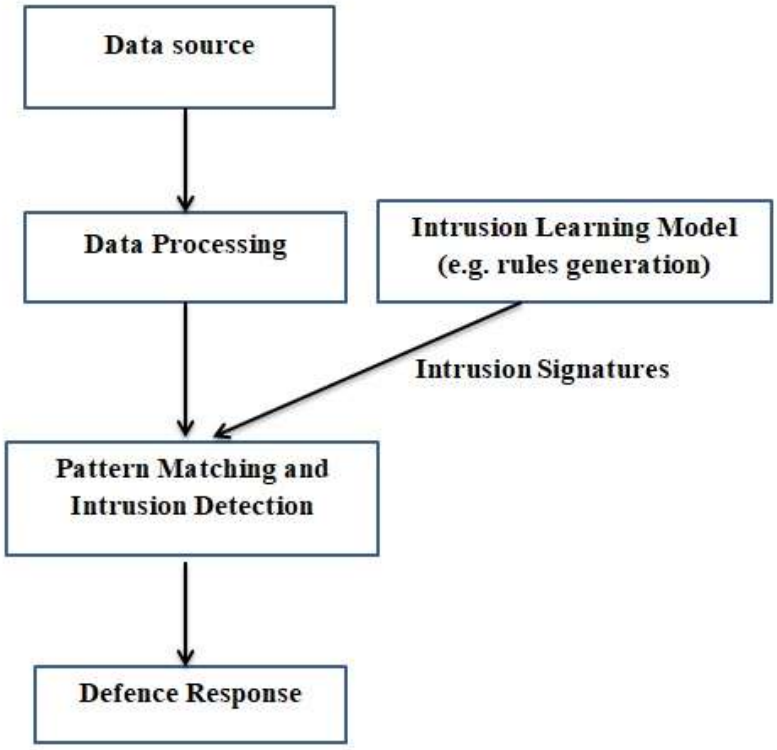

Figure 2.1: Sequence of execution of signature detection modules adopted from [15]

The data is gathered from network and host audit logs, packets transmitting over the network, and windows log and registry. Data pre-processing [15] is a critical step that prepares the raw data for learning patterns. It is involved normalizing or ordering of data, the reduction of noise by eliminating outliers, and finally selecting and extracting features. After the data pre-processing, machine learning system is deployed to build a learning model and extract rules using prior knowledge of the execution of malicious programs, network traffic data, and vulnerabilities in network infrastructure. The model is now ready for signature and misuse detection. The learned classification model is applied to the incoming network traffic for signature detection. If any part of the network traffic is found to be similar to attack patterns

learned by the model, then an alarm is raised and the traffic is further analyzed for identifying whether it is really an attack or a false alarm.

The design idea behind anomaly detection IDS [4] is to establish a normal behavior profile and then define abnormal behaviors by their degree of deviation from the normal profile. Thus, the key to designing an anomaly detection system is to clearly define a normal profile. The benefits of anomaly detection are strong generalizability, ability to identify zero-day attacks and the ability to recognize unknown attacks due to the fact that recognizing the abnormal user activity does not rely on a signature database, while its shortcomings are a high false alarm rate and an inability to provide possible reasons for an abnormality [4].

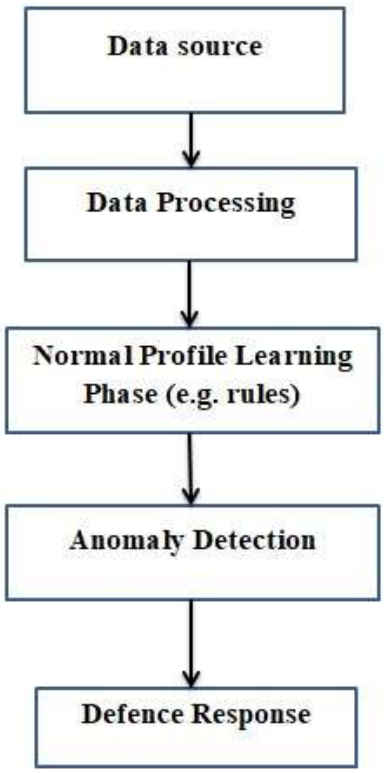

Figure 2.2: Sequence of execution of modules in anomaly detection adopted from [15]

In Figure 2.2 explained that anomaly detection has five steps [15]. As we have known a machine learning phases are involved, like data gather from network, host or both. Also data processing includes the volume of data is reduced as this step includes feature selection, feature extraction, and finally dimensionality reduction processes.

The normal profiling learning step [4] is learning a normal behavior from the data. At anomaly detection is conducted identification of abnormal behaviors using dissimilarity detection techniques. Finally, it responds to classified normal from abnormal profile and responds the alert.

Intrusion Detection Systems [8] are security systems that collect information from various types of system and network sources, and analyze these data in an attempt to detect activity that may constitute an attack or intrusion on the system. Usually, the attacks target not only one individual computer but also aim for a group of hosts. As a result, some intrusions might show an anomalous behavior at the network layer, while others could exhibit anomalous behaviors at the application layer.

IDS is classified based on data source-based methods depending upon the origin of data source or location in a network, such as network packets, payload, operating system logs ,firewall logs and network sensors as shown in Figure 2.3. There are two types IDS as host based IDS and Network-based. Host-based IDS (HIDS) is an installed software package which monitors a single host for suspicious activity by analyzing events occurring within that host [16].

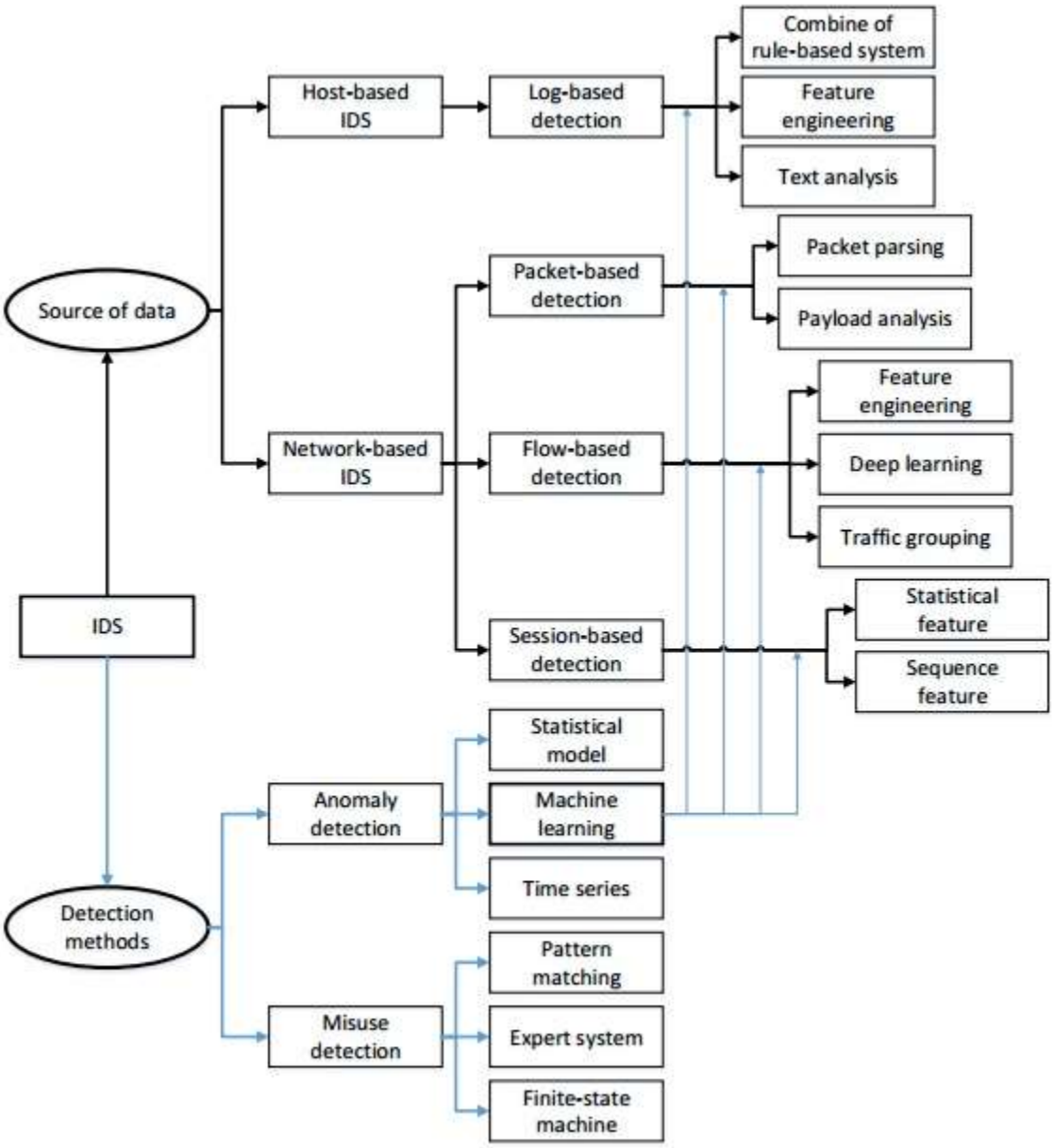

Figure 2.3: Classifications of IDS adopted from [4]

HIDS is usually software running on the protected host, and therefore they must be installed on each individual machine and requires configuration specific to that operating system. Some of the advantages are monitor all users' activities, identifying attacks that originate from inside the host and analyze the decrypted traffic to find attack signature thus giving them the ability to monitor encrypted traffic. In other hand, the disadvantages are that host-based IDSs occupy host resources as a major storage space and extra computing power from the host where they reside.

They can be ineffective during the denial of service attacks. Also they are dependent on the reliability of the host, and are unable to detect network attacks.

Network –based IDS (NIDS) [17] is monitor network traffic data using a set of sensors attached to the network to capture any malicious activities. Networks security problems can vary widely and can affect different security requirements including authentication, integrity, authorization, and availability. A majority of network-based IDSs are independent of the operating system (OS); thus, they can be applied in different OS environments [4].

Furthermore, network-based IDSs are able to detect specific types of protocol and network attacks. There are advantages of NIDS; monitors only read each packet pass through a network segment without taking computing resource from host. Also they can be implemented easily on part of a network and independent from operating system. However, there are disadvantages of NIDS. Due to monitoring of every packet that passed through the segment, they have difficulty keeping up with systems especial systems with heavy traffic. On this study, we emphasize on anomaly detection on network-based intrusion detection which is discussed in the next sub-section.

2.2. Anomaly Intrusion Detection

Network inspired by nervous system has become an interesting tool in the applications of Intrusion Detection Systems. It supports an ideal specification of an Intrusion Detection System and is a solution to the problems of traditional IDSs [18] Anomaly-based detectors attempt to estimate the ‘‘normal’’ behavior of the system to be protected, and generate an anomaly alarm whenever the deviation between a given observation at an instant and the normal behavior exceeds a predefined threshold [19]. These profiles are constructed from historical data collected during normal operation.

The detectors collect data from the events and use a variety of measures to determine when the monitored activity deviates from normal activity [14]. However, due to the assumptions underlying anomaly detection mechanisms, their false alarm rates are in general very high. Specifically, the main reasons for this limitation includes the user’s normal behavior model is based on data collected over a period of normal operations; intrusive activities missed during this period are likely to be considered as normal behaviors. Also

anomaly detection techniques can hardly detect stealthy attacks because these kinds of attacks are usually hidden in large number of instances of normal behaviors.

Moreover, the types of parameters used as inputs of normal models are usually decided by security experts. Any mistake occurring during the process of defining these parameters will increase the false alarm rate and decrease the effectiveness of the anomaly detection system. As a result, the design of the detection methods and the selection of the system or network features to be monitored are two of the main open issues in anomaly detection [14].

ADIDS [8] methods can be categorized into three main groups: Statistics-based, knowledge based and machine learning-based. ADIDS is established on the statistical analysis , it detects attacks based on abnormalities in the pattern with respect to the normal pattern of data on the network, only selects the normal states from the pattern as usual activity and the rest as attack on the network.

Statistical-based techniques use statistical properties such as mean and variance on normal transaction to build the normal profile [20]. It detects possible system intrusions by identifying departures from historically established normal behavior. The statistical tests are employed to determine whether the observed transaction deviates from the normal profile. The IDS assigns a score to the transactions whose profile differs from the normal. If the score reaches the threshold, alarm is raised. The threshold value is set based on count of events that occur over a period of time.

Knowledge-based techniques are used to extract the knowledge from the specific attacks and system vulnerabilities. This knowledge can be further used to identify the intrusions or attacks happening in the network or system. They generate alarm as soon as an attack is detected. They can be used for both misuse and anomaly-based detection [8].

Machine learning (ML) strategies emphasize on building a framework that enhances its execution based on previous results, it can change their execution strategy based on recently acquired data. ML model does not learn through a database of labeled attacks with known patterns and signatures but rather uses features of network traffic flow such as source address, destination address, bytes per flow, source port, destination port, and much more to learn the general feature set of normal traffic [21] . These features are monitored over a period of time and

are used as the dataset to train the ML model. Using machine learning algorithms can effectively improve the accuracy of detection and reduce the requirement of human knowledge.

Machine learning is divided into shallow learning and deep learning. Shallow learning relies on a field expert to identify the relevant features for evaluation. Thus, among the features that are in a flow, the field expert must select the features they believe are the most relevant and use that to train the ML model. In addition, Principal Component Analysis (PCA) can be used as feature selection method. On the other hand, Deep learning relies on the model selecting the features it estimates to be the most important ones for determining the effectiveness of the model without intervention of expert knowledge. The common machine learning algorithms used in IDSs are shown in Figure 2.4.

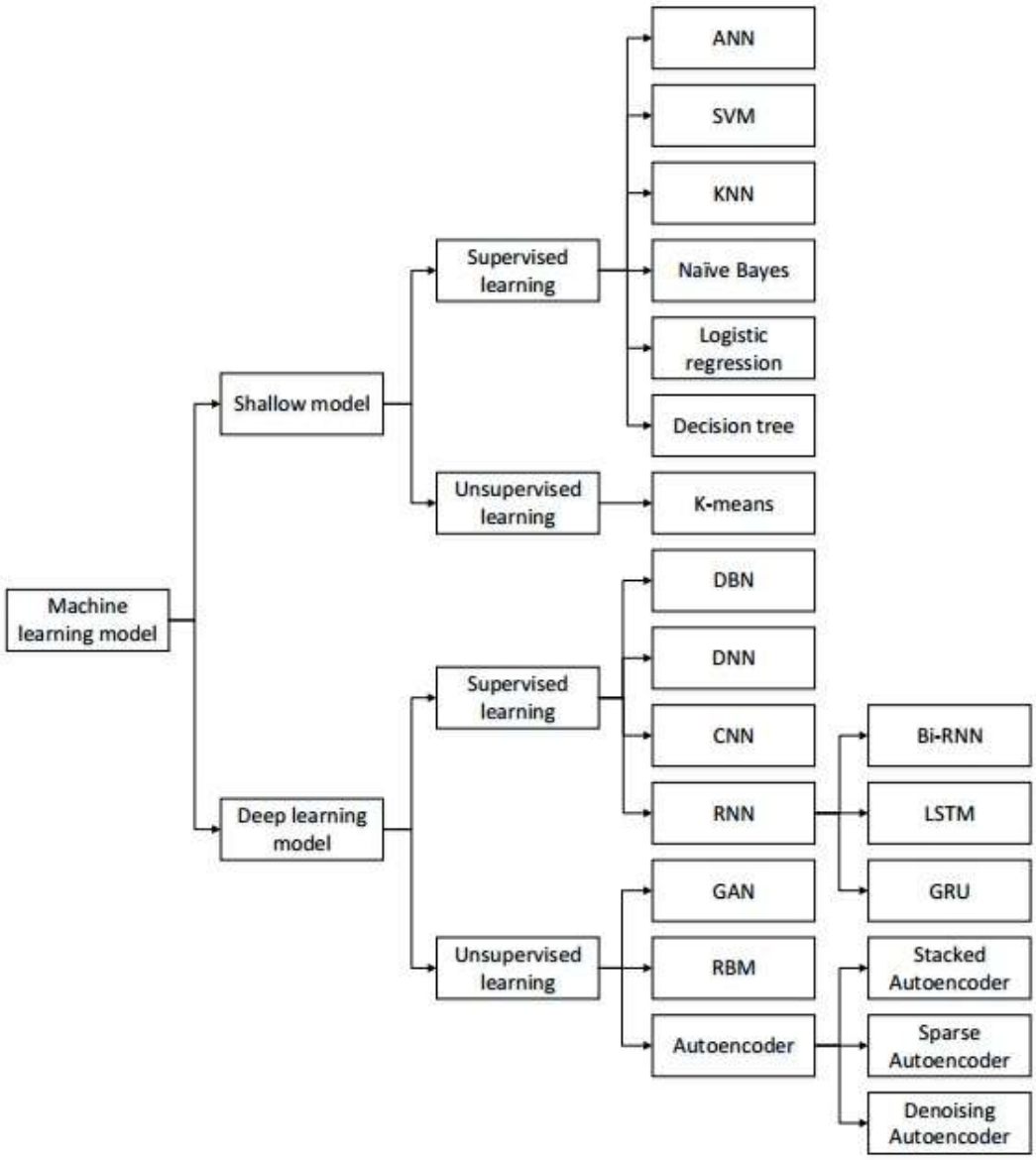

Figure 2.4: Taxonomy of machine learning algorithms adopted from [4]

2.3. Deep Learning in Intrusion Detection

In recent years, security researchers are more focus on the application of deep learning in the different areas. IDS are one of application area in order to strength security aspects on the networks and systems. Neural networks [19] have been adopted in the field of anomaly intrusion detection, mainly because of their flexibility and adaptability to environmental changes. This detection approach has been employed to create user profiles, to predict the next command from a sequence of previous ones, to identify the intrusive behavior of traffic patterns.

The major variance between shallow learning and deep learning is that deep learning design has multiple hidden layers. Feature selection can be performed by the first few layers of deep neural network, which enables deep learning to extract advanced features so that high-level concepts can be learned, which makes up for the defects of machine learning algorithm. In addition, depending on how the techniques can be used, deep learning can be categorized into: (1) deep networks for unsupervised learning (2) deep networks for supervised learning.

Supervised learning [14] is the task of gathering a function from labeled training data. The training data consist of a set of training examples. The learner receives those sets of labeled examples as training data and makes predictions for all unseen points. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Supervised learning problems are commonly associated with "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Supervised learning is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the predetermined classifications. In the case of neural networks, the classification is used to determine the error of the network and then adjust the network to minimize it.

In addition, IDS based on supervised learning can use feature selection to exclude unnecessary features in the training data, and use the remaining selected features to train the classifier to learn the internal relationship between input data and labeled output values [8]. The supervised machine learning algorithms that have been applied to intrusion detection using deep learning

including: Deep Convolutional Neural Network (DCNN) [22] Convolutional Neural Network (CNN) [23], Recurrent Neural Network (RNN) [24] and Deep Belief Network (DBN) [25].

A supervised learning approach usually consists of two stages [8], namely training and testing. In the training stage, relevant features and classes are identified and then the algorithm learns from these data samples. In supervised learning IDS, each record is a pair, containing a network or host data source and an associated output value namely intrusion or normal. Next, feature selection can be applied for eliminating unnecessary features. Using the training data for selected features, a supervised learning technique is then used to train a classifier to learn the inherent relationship that exists between the input data and the labeled output value. In the testing stage, the trained model is used to classify the unknown data into intrusion or normal class. The resultant classifier then becomes a model which, given a set of feature values, predicts the class to which the input data might belong.

Unsupervised learning is a form of machine learning technique used to obtain interesting information from input datasets without class labels. [8] It creates joint density models from a set of random variables without class labels and obtains useful information from them. The label of the output data in supervised learning IDS is given and used to train the model to handle the unknown data, while in unsupervised learning IDS the label is unknown, and instead of that, the data is automatically divided into different classes during the learning process. Normal records will form sizable clusters, and the records in other small clusters will be labeled as malicious attack data, because the performance of malicious records and normal records is not the same, so they belong to different clusters.

The main downside of supervised learning in deep learning intrusion detection is the need of tagging the training data, which makes the process costly, time consuming and challenging to find new attacks. In other hand, unsupervised learning addresses these matters solve by training based on unlabeled datasets as only input data is given, finds all kind of unknown patterns in data, so all the input data to be analyzed and labeled in the presence of learners and therefore facilitating operational learning and improving detection accuracy through the detection process.

Unsupervised learning algorithms are relatively new than supervised to work on intrusion detection systems. Unsupervised learning algorithms that have been applied to intrusion

detection using deep learning include: Autoencoder [26], [27], Restricted Boltzmann Machine (RBM) [26], [28], [29] and Generative Adversarial Network (GAN) [30], [31].

There are various unsupervised deep learning algorithms, among which the Generative Adversarial Networks (GAN) used in this research is the most promising one. GAN is first introduced in 2014 [7]. GAN has shown great results in many generative tasks to replicate the real-world rich content such as images, human language, and music. It is inspired by game theory: two models, a Generator and a Discriminator, are competing with each other while making each other stronger at the same time.

The two models, the Generator and Discriminator, are trained together [32]. The Generator generates a batch of samples, and these, along with real examples from the domain, are provided to the Discriminator and classified as real or fake. The Discriminator is then updated to get better at discriminating real and fake samples in the next round, and importantly, the Generator is updated based on how well, or not, the generated samples fooled the Discriminator. After the training process, the Generator model is discarded as we are interested in the Discriminator.

The Discriminator model takes an example from the domain as input (real or generated) and predicts a binary class label of real or fake (generated). The real example comes from the training dataset. The generated examples are output by the Generator model.

In supervised learning, we may be interested in developing a model to predict a class label given an example of input variables. This predictive modelling task is called classification. Classification is also traditionally referred to as discriminative modelling.

2.3.1.Generative Adversarial Networks

Generative models attempt to learn the exact distribution of real data for modeling, and their importance is significantly increased because of their high adaptability in various fields. However, most of the traditional generative models use the maximum likelihood principle to train the model, in order to make the parameterization of the model approximate to the real data distribution as much as possible, which makes these models inadequate in dealing with the complexity of high-dimensional data.

A GAN model includes two sub networks; these are Generator G and a Discriminator D. The Generator aims to generate synthetic data similar to the real data, and the Discriminator intends to distinguish artificial data from real data [8]. Thus, the Generator and the Discriminator improve each other. GANs are currently a hot research topic used to growth data in attack detection, which partly ease the problem of IDS dataset shortages. Meanwhile, GANs belong to adversarial learning approaches which can raise the detection accuracy of models by adding adversarial samples to the training set.

According to the theoretical descent from Goodfellow, after several steps of training and assuming that G and D both have sufficient capacity, the real data probability distribution will be the same as the data probability distribution provided by G, and neither G nor D can be improved, that is, when the optimizations achieved, an equilibrium state will occur between G and D, and D's output is 0.5 [7]. Two points can be inferred from this derivation. First of all, GAN can solve the likelihood difficulty with only using the relative behavior of the two distributions. Secondly, GAN can measure the inconsistency between the generated data distribution and the real data distribution in an implicit way through D, and then learn to reduce the inconsistency.

However, still there are problems on standard GAN. In order to generate high-resolution and high-quality samples, both the Generator and Discriminator are asked to be deeper and larger. Under the exposed adversarial framework, it’s hard to balance and optimize such large-scale deep networks [33]. So it has to be appropriate hyper-parameters like learning rate, updating steps and network architectures are critical configurations. Unsuitable settings reduce GAN’s performance or even fail to produce any reasonable results. Since the training strategy is fixed, it is hard to adjust the balance between the Generator and Discriminator during the training process fine tune training as shown in Figure 2.5.

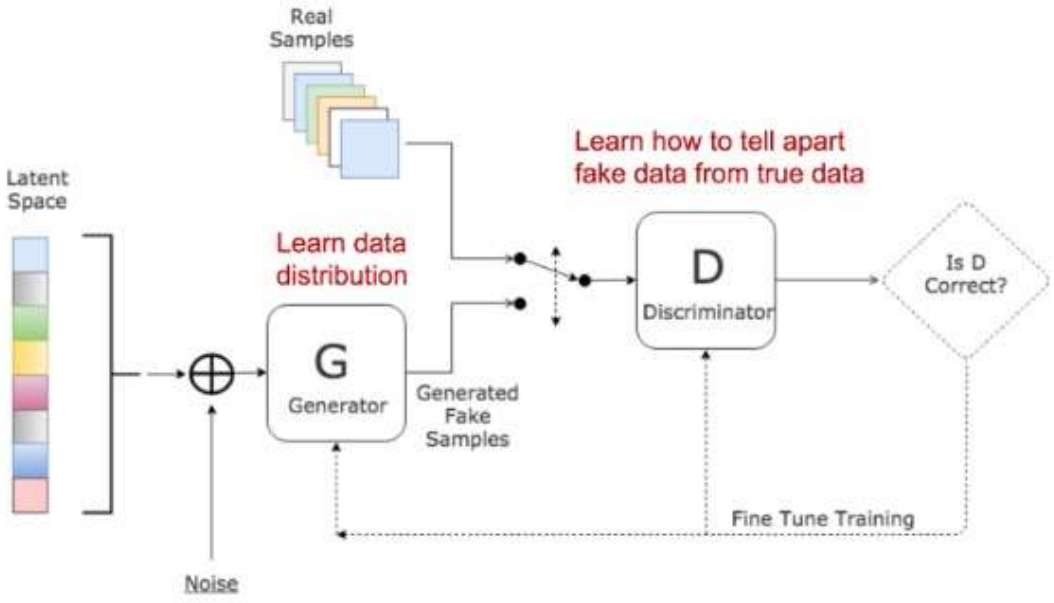

Figure 2.5: Architecture of a generative adversarial network adopted from [34]

There are many researches developed various GAN variants to solve training instability by changing the objective function, the structure, etc. Typically, assuming the optimal Discriminator for the given Generator is learned, different objective functions of the Generator aim to measure the distance between the generated distribution and the target data distribution under different metrics.

The original GAN uses Jensen-Shannon divergence (JSD) as the metric [35]. A number of metrics have been introduced to improve GAN's performance, such as least-squares [36], Kullback Leibler (KL) divergence [37] and Wasserstein distance [38]. However, according to both theoretical analyses and experimental results', minimizing each distance has its own pros and cons. For example, during the training iteration is increases or number of epoch, the Generator may collapse where it always produces same output. The model is measured KL divergence that eliminates the vanishing gradient issues [38], [39]. According to the paper released [40], if the dimensionality of noise is less than the dimension of the real dataset (as is typically the case), then it's impossible for Generator to be continuous. These are because in most cases Generator outputs will be contained in a union of low dimensional manifolds, and

therefore have measure 0 in original dataset. Likewise, Wasserstein distance greatly improves training stability but can have non-convergent limit cycles near equilibrium [41].

2.3.2.Wasserstein Generative Adversarial Networks

The Wasserstein GAN (WGAN) was introduced in 2017 paper titled "Wasserstein GAN". It is one of variance of GAN that seeks an alternate way of training the Generator model to better approximate the distribution of data observed in a given training dataset. Instead of using a Discriminator to classify or predict the probability of generated data as being real or fake, the WGAN changes or replaces the Discriminator model with a critic that scores the realness or fakeness of a given data [38].

The WGAN remedies the problem by redefining the loss function with Wasserstein distance, which makes the training process stable and less sensitive to hyperparameter selection. WGAN attempts to minimize the distance between the distribution of the data observed in the training dataset and the distribution observed in generated data; which is also called the earth mover distance (EM).

EM distance can provide a meaningful and smooth representation of the distance between two distributions even in the case without overlaps. Given the competitive nature between Generator and Discriminator, we don't have a clear point at which we want to stop training. Also it measures the distance horizontally and similarity between both probability distributions unlike Kullback Leibler and Jeson Shenon divergence measures vertically on original GAN.

The WGAN [35] uses the earth mover's distance as a loss function that clearly correlates with the visual quality of the samples generated. The benefit of the WGAN is that the training process is more stable and less sensitive to model architecture and choice of hyperparameter configurations.

2.4. Overfitting and Underfitting

In Deep learning or other machine learning approaches, models are trained very well and correct predictions for any sample in the training data, however can't make correct predictions for unseen data outside the training samples. Model is considered right when it behaves nearly same way on training and test data with highest accuracy. However if we did not get the predictions as expected it will cause the problem on performance of the model. Now, suppose we want to check

how well our machine learning model learns and generalizes to the new data. For that we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

2.4.1. Overfitting

Overfitting happens when a model learns the details and noise in the training data which can negatively impact the performance of the model on new unseen data. When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. Overfitting can be prevented using different strategies such as cross-validation, feature selection, reducing network size in neural network, dropout and early stopping [42].

2.4.2. Underfitting

Underfitting happens when algorithm used to build prediction model is very simple and not able to learn complex pattern from the training data. In that case accuracy will be on lower side on seen training data as well as unseen test data. Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data. In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.

2.5. Hyperparameter Tuning

Hyperparameters are important for machine learning algorithms since they directly control the behaviors of training algorithms and have a significant effect on the performance of machine learning models [43]. It is a parameter whose value is used to control the structure of neural network and during training and testing of model. Hyperparameters are variables set by the data scientist before training and helps to control the implementation of the model. For example, for

deep learning algorithms, the following are initialized hyperparameters network weight initialization, activation function, number of hidden layers and units, learning rate, momentum, number of epochs and batch size. Models can have various hyperparameters and finding the best combination of parameters can be treated as a search problem. Three strategies for hyperparameter tuning are [44]: grid search, random search and Bayesian optimization.

2.5.1. Grid Search

Grid search is the most basic hyperparameter tuning method. With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the optimal results.

2.5.2. Random Search

Random search differs from grid search in that we longer provided a discrete set of values to explore for each hyperparameter; rather, we provide a statistical distribution for each hyperparameter from which values may be randomly sampled.

2.5.3. Bayesian Optimization

The previous two methods performed individual experiments building models with various hyperparameter values and recording the model performance for each. Because each experiment was performed in isolation, it is very easy to parallelize this process. However, because each experiment was performed in isolation, we cannot able to use the information from one experiment to improve the next experiment.

2.6. Related Works

There has been done a lot of research on intrusion detection. Researchers working on IDS based on machine learning algorithms have also developed many methods to improve system performance. Niyaz has research on both SIDS and ADIDS using self-taught learning, a deep learning technique based on sparse auto-encoder and soft-max regression, to develop an NIDS. STL has two stages for classification which are Unsupervised Feature Learning (UFL) on unlabeled data and classification on labeled data [9].

The experiment was conducted on NSL-KDD dataset training data without labels for feature learning using sparse auto encoder for the first stage of self-taught learning. In the second stage they apply the newly learned features representation on the training data itself for the

classification using soft-max regression. It was evaluated the performance of STL and SM using the test data and STL perform better in anomaly detection than SM. But from the research they only uses one dataset may have impact differently on both algorithms due to the unbalanced data properties.

According to the paper [45] using different deep neural network architectures including Convolutional Neural Networks, Autoencoders, and Recurrent Neural Networks. These deep models were trained on NSLKDD training dataset and evaluated on both test datasets provided by NSLKDD namely NSLKDDTest+ and NSLKDDTest21. For training and evaluation of deep models, a GPU powered test-bed using Keras with Theano backend was employed. To make model comparisons more credible, the researchers implemented conventional ML IDS models with different well-known classification techniques including Extreme Learning Machine, k-NN, Decision-Tree, Random Forest, Support Vector Machine, Naive-Bays, and QDA. Both DNN and conventional ML models were evaluated using well known classification metrics including ROC Curve, Area under ROC, Precision-Recall Curve, mean average precision and accuracy of classification. Both DCNN and LSTM models showed exceptional performance with 85% and 89% Accuracy on test dataset.

In the paper proposed by Farahnakian used four Autoencoders where the output of each auto-encoder in the current layer is used as the input of the auto-encoder in the next layer [27]. In addition for classify purpose; softmax classifies the attack classes from the input dataset and full fining to find optimal hyper-parameter. The experimental results show low false negative rate (0.42%), high accuracy (94.71%) and high detection rate (94.53%) on KDD-CUP'99 dataset. However still a classifier used for detect unknown attack is limited to domain knowledge.

In the research [46], GAN is used to improve the malware detection effect. To evade detection, malware applications try to generate packets similar to normal packets. They configured a virtual network system with hosts, servers, and an IPS. They have used private dataset for the study to show GAN improve the detection for unknown attacks. To collect the dataset, they started up trained a GAN model. The GAN guided the malware to produce packets similar to Facebook. As the training epochs increased, the packets blocked by the IPS decreased and packet that passed inspection increased. The result was that the malicious packets generated by the GAN were more similar to normal packets. Then, by analyzing the generated packets, the robustness of the IPS

was improved. But still the study do not showed the evaluation metrics for in which percent false alarm rate decreased and accuracy achieved.

The paper has elaborately summarized the usefulness of DNNs in IDS. For the purpose of reference, other classical ML algorithms have been accounted and compared against the results of DNN. The experiment is conducted on Keras with TensorFlow with GPU enabled on Nvidia-GK110BGLTesla-k40. Also, hyper-tuning of parameters to figure out the optimum set of parameters to achieve the desired result is all by itself a separate field with plenty of future scope for research. The paper used the learning is kept constant at 0.01 while the other parameters where optimized. The count of the neurons in a layer was experimented by changing it over the range of 2 to 1024. After that, the count was further increased to 1280 but did not yield any appreciable increase in accuracy. Therefore the neuron count was tuned to 1024. Additional optimization used on selecting 5 counts of hidden layers models in which layers of 1, 2, 3, 4 and 5 respectively for DNN. Different datasets have reviewed and KDDCup-'99' was selected to work on. The publicly available KDDCup-'99' dataset has been primarily used as the benchmarking tool for the study, However in the data processing , the research don not take account the categorical features and only taken 41 features as input. Finally, the best performance was showed by DNN consisted 3 hidden layer compared to all the others and measured Accuracy 93.0 %, Precision 99.7, Recall 91.5 and F1 score 95.5% [47].

According to the paper [48] proposed in framework focuses on incorporating deep adversarial learning with statistical learning and exploiting learning based data augmentation with a GAN. The KDD99 dataset is both unbalanced and lacks new data, which leads to poor generalizability of machine learning models. To address these problems, they utilized a GAN to expand the dataset. The GAN model generated data similar to the flow data of KDD99. Adding this generated data to the training set allows attack variants to be detected. They selected 8 types of attacks and compared the accuracies achieved on the original dataset compared to the expanded dataset. The experimental results showed that adversarial learning improved 7 accuracies in 8 attack types [48].

The study have been done by Liu [4], to identify the challenges in the IDS's. Also it is a baseline for this research paper. Although machine learning methods have been great advances in the field of intrusion detection, the following challenges were identified. The first one is lack of

available datasets, in the field of computer security it is hard to find dataset more resemble the network implementation environment due to security risk and privacy issues [4]. There are limited publicly available dataset such as KDD99, NSL-KDD, DARPA and ISCX 2012 [8]; each of them have its own strength and weakness. But the main problem is shortage of amount of tagged and identifies anomaly behaviors. On the other hand, constructing new datasets depends on expert knowledge, and the labor cost is high. New types of attacks are emerging, and some existing datasets are too old to reflect these new attacks.

Secondly, most studies emphasize the detection results tends to low efficiency; therefore, they usually employ complicated models and extensive data preprocessing methods, leading to low efficiency [4]. However, to reduce harm as much as possible, IDSs need to detect attacks in real time. Thirdly, lower detection accuracy in actual environments which IDS have implemented based on ML a certain ability to detect intrusions, but they often do not perform well on completely unfamiliar data. The dataset does not cover all typical real-world samples. A lot of research works have been done which can be applied to intrusion detection. But the related works we reviewed showed the performance, detection rate, training time, and False Alarm Rate in IDS remains one of the major issues.

In order to overcome the above challenges, IDS have been developed based on ML. The major trends of IDS research focused on the following aspects [4].

1. Combining domain knowledge with machine learning can improve the detection effect, especially when the goal is to recognize specific types of attacks in specific application scenarios. Which it is rule-based and hybrid approaches.
2. Improvements in machine learning algorithms are the main means to enhance the detection effect. Thus, studies involving deep learning and unsupervised learning methods have an increasing trend.

In general, the Network Security applications of ML [46] involve spam classifiers, malware analysis, and intrusion detection and network traffic identification. Table 2.1 summarizes the application domains that have been studied in the past. As it can be seen, there are few studies related to Intrusion detection and they both assume an active influence model.

Table 2.1: Security applications of adversarial ML

Application	References
Spam Bayes classifier	Joseph [49] , Zhang [48]
Malware analysis	Grosse [50]
Anomaly Intrusion detection	Ezeme [51], Niyaz [9]
Traffic identification	Ateniese [52]

Generative adversarial networks is one of unsupervised deep learning algorithm which is based on game theory by generating samples with Generator and identify normal profile from generated samples by Discriminator [7]. It solved the shortage of datasets as well as the Discriminator more learning from fake generated samples that tries to fool it. However there are factors that limited affect the efficiency of the algorithm and the variance effects on IDS have to be studied deeply [38].

CHAPTER THREE

METHODOLOGY

3.1. Overview

In this study, the research approach is followed design science and presented the design of the proposed intrusion detection. Different components of the proposed IDS are described with their relevance and techniques to use while building those components. It includes research design, type and sources of data, the tools and techniques, algorithms, data analysis and presentation, and evaluation metrics.

This chapter summarizes the components of IDS based on GAN, including: data pre-processing, IDS structure model, detection methods, and evaluation metrics. Data pre-processing describes how to numeric convert and normalize the data in the NSL-KDD dataset so that the data becomes vectors of completely numeric converted that can be entered into IDS. IDSes based on GAN and WGAN are implemented by using the programming language Python with Keras as the deep learning framework and tensor flow as backend.

3.2. Research Design Approach

In this work, we followed a design science approach to mitigate the unknown attack using a deep learning neural network model. The research design was comprised by the result of the literature review. Firstly it is conduct a survey on review literature to acquire a general understanding of the research area and come up with specific topic to be review comprehensively. This brings a deeper understanding about a topic and its problems domain to be clearly revealed. Through this literature we identify the importance of the previous works conducted by different scholars in the area of intrusion detection system [4], [8]. Existing works related to this research work weighed to identify and point direction in providing solution to identified problems.

The second thing to do is to set objectives to be accomplished in order to solve problem and scope of the research to be touched and focused. Thirdly, the data should be collect from different datasets to study patterns that can identify known and unknown attacks from normal data. In the train-test split phase the whole preprocessed data is split into training set for training deep learning algorithms and testing set for evaluating models' performance. In the validation

split phase, the training set is further split into actual training and validation set. In training with default parameters phase, the machine learning algorithm is trained using the training data on their default parameters and produced a trained model. The trained model is then evaluated its performance on the separated preprocessed testing data.

On the other hand, the grid search phase is used to find the optimal parameters for a model by constructing a parameter grid. In this phase first the training and validation data is passed to the grid search. Then, the parameter grid of hyperparameters and classification algorithm are specified and passed to the cross-validation. The cross-validation searches the whole parameter grid and running the training set for selecting the best parameters. Then, the models are evaluated on the validation set. The selected best parameters are then used for building final trained model which is then evaluated on the testing set.

Next, we have proposed a solution from the identified problem on the literature survey; we have selected GAN and WGAN algorithms used. After identifying those optimal hyperparameter requirements, we designed and implement of Anomaly Intrusion detection models using GAN and WGAN algorithms. Then finally we evaluated the model using test dataset based on IDS metrics like accuracy, precision, recall, accuracy and F1-score. [53].

3.3. Data Preparation

In this preprocessing phase, the raw dataset of NSL-KDD intrusion detection dataset is downloaded from Internet based on the paper [54] Canadian Institute for Cyber security organization. The datasets used for network packet analysis in commercial products are not easily available due to privacy issues. However, there are a few publicly available datasets such as DARPA, KDDCup 1999, NSL-KDD and ADFA-LD and they are widely used as benchmarks. To the best of our knowledge, two datasets such as KDDCup 1999 dataset and NSL-KDD dataset have been popularly employed as training and testing datasets. In order to conduct this research, the researcher used secondary sources of data [55] which previously being collected through network. This offline dataset is used in the training phase of this module. It is labeled dataset that can easily learn the system. In this study, we used simulated dataset called NSL-KDD for this training and testing phase [55].

3.3.1. Dataset Description

The NSL-KDD dataset is refined version of the KDD'99 dataset. The KDD cup dataset has been widely used as a benchmark dataset in NIDS evaluation for many years, but it has a serious defect that both the training data and the test data contain a large number of redundant records. In the training dataset and the test dataset, approximately 78% and 75% of the records were redundant, respectively [54]. These redundant records make the learning algorithm biased against frequent attack records but neglects infrequent harmful attack records. And the large numbers of frequent normal records also decline the algorithm's learning of attack records.

The NSL-KDD dataset eliminates all redundant records in the training and test data of the original KDD'99 dataset, and replaces the KDD'99 dataset as the benchmark dataset for NIDS evaluation. Records of different classes are balanced in the NSL-KDD, which avoids the classification bias problem. The NSL-KDD [4] also removed duplicate and redundant records; therefore, it contains only a moderate number of records. Therefore, the experiments will be implementing on the whole dataset.

Number of datasets available in NSL-KDD which consist of two parts [13]: (i) KDDTrain+ and (ii) KDDTest+. The KDDTrain+ part of the dataset NSL-KDD is used to train a system to detect network intrusions or the detection method. It consists of four classes of attacks and a normal class data set. The KDDTest+ part of NSL-KDD dataset is used for testing a detection method or a system when it is evaluated for performance. The training is performed on KDDTrain+ data which contain 22 attack types and testing is performed on KDDTest+ data which contains additional 17 attack type.

Table 3.1: Records distribution of training set and testing set in NSL-KDD dataset

	Total	Normal	Attack				
			Dos	Probe	R2L	U2R	Total Attack
KDDTrain+	125972	67342	45927	11656	995	52	58630
KDDTest+	22542	9710	7458	2421	2887	67	12832

NSL-KDD dataset comprises close to 4,898,431 unique connection vectors, where every connection vector consists of 42 features of which 34 are continuous features and 7 are discrete features and 1 class attribute. These 42 features are included three types: Binary, Numeric and Nominal. Each vector is labeled as either normal or attack. There are four major categories of attacks labeled in NSL-KDD: denial of service, probing, users-to-root and remote-to-local attack. Totally it includes 5 classes with normal attack class [56].

A) **Denial of Service Attack (DoS):** is an attack in which the attacker makes some computing or memory resource too busy or too full, achieved by flooding target hosts or networks to handle legitimate requests, or denies legitimate users access to a machine e.g. synchronize flooding, Relevant features: "source bytes" and "percentage of packets with errors".

B) **Probing Attack:** is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls. The attacker probes various ports in order to observe what subset of traffic is being monitored at which IP addresses, which enables the attacker to target subsequent pernicious activity toward those less tightly managed hosts e.g. port scanning, Relevant features: "duration of connection" and "source bytes".

C) **Remote to Local Attack (R2L):** occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access that would enable the attacker to exploit a local user's privileges as a user of that machine e.g. password guessing, Relevant features: Network level features – "duration of connection" and "service requested" and host level features: "number of failed login attempts".

D) **User to Root Attack (U2R):** Initially attacker access normal user account, later gain access to the root by exploiting the vulnerabilities of the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system. Table 3.2 describes a list of NSL-KDD features in dataset. E.g. buffer overflow attacks, Relevant features: "number of file creations" and "number of shell prompts invoked".

Table 3.2: NSL-KDD features and value types adopted from [10]

Types	Features
Nominal	Service(3), Protocol_Type(2), Flag(4)
Binary	Su_Attempted(15), Is_Host_Login(21), Root_Shell(14), Is_Guest_Login(22), Land(7), Logged_In(12)
Numeric	Duration(1), Dst_Bytes(6), Urgent(9), Src_Bytes(5), Count(23), Num_Failed_Logins(11), Num_Root(16),Hot(10), Num_Access_Files(19), Wrong_Fragment(8), Rerror_Rate(27), Dst_Host_Srv_Serror_Rate(39), Dst_Host_Srv_Count(33), Srv_Diff_Host_Rate(31), Srv_Count(24), Num_File_Creations(17),Dst_Host_Diff_Srv_Rate(35), Num_Shells(18), Num_Compromised(13), Dstdst_Host_Rerror_Rate(40), Num_Outbound_Cmds(20), Serror_Rate(25), Dst_Host_Count(32), Dst_Host_Same_Srv_Rate(34), Diff_Srv_Rate(30), Srv_Rerror_Rate(28), Same_Srv_Rate(29), Dst_Host_Serror_Rate(38), Dst_Host_Same_Src_Port_Rate(36), Srv_Serror_Rate(26), Dst_Host_Srv_Diff_Host_Rate(37), Dst_Host_Srv_Rerror_Rate(41),

The 42nd attribute contains data about the various 5 classes of network connection vectors and they are categorized as one normal class and four attack class shown in Table 3.3.

Table 3.3: Mapping of attack class with attack type

Attack Class	**Attack Type**
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm (10)
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

3.3.2. Data Processing

Data preprocessing phase is one of the critical steps in data mining process which performs the preparation and transformation of the original dataset. The various steps are included in data processing [57]; they are data cleaning, feature reduction and future construction. Feature reduction includes feature extraction and feature selection. However, in deep learning feature reduction is conducted automatically by neural network.

Machines need specific representations of the input data for both training and testing which is termed features before feed to a deep learning algorithm. In order to prepare the data in that format preprocessing tasks such as string indexing, one-hot encoding, feature selection, and normalization are performed on the training and testing data.

The first task is done by converting both KDDTrain+ and KDDTest+ .text to .csv file and NSL-KDD dataset contains 38 numeric features and 4 non-numeric features as attribute [58]. We have done transforming of each symbolic column of the data to numeric. Each column has its own customization table, depending on the column content. Non numeric attribute like protocol type, service and flag attribute need to convert as numeric attribute using LabelEncoder algorithm, because the training input and testing input is given to GAN and WGAN should be numeric matrix. So these categorical features are identified and extracted to 3 protocol type, 70 service and 11 for flag in total 84 dummies column formed using One-Hot-Encoding. Continuing in this way, 42-dimensional features map into 84 categorical features to get 126 dimensional features after transformation. After that it is dropping the three categorical features and difficulty feature which are expanded to get 122 features dimensional dataset.

The class feature also labelled as numeric type, to do this arrangement all values is categorized into 5 classes starts with normal labelled as 0, DOS labelled as 1, R2L labelled as 2, U2R labelled as 3 and probe labelled as 4. However in this study only focused on 2 –class classification normal and attack. So the dataset is re-arranged for 2-class 1 for normal and 0 for attack.

Feature scaling is an essential step to deal with local optima, and skews towards particular features. It also facilitates the ML-based IDS with faster training. We apply standard scaling MinMax Scaler, which it is transform features by scaling each feature to a given range. Also the

difference in all features value to be scale between 0 and maximum value in the dataset in order to flatten more or less look like standard normally distributed data.

According to above five classifications are converted to two classes with 0 for attack and 1 for normal transaction in the dataset. Also the dataset is prepared in to two formats. The first one is included only normal transactions and the second one is both attack and normal transactions are included shown in Figure 3.1. In this step, the train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data (dataset which has normal and attack class) not used to train the model.

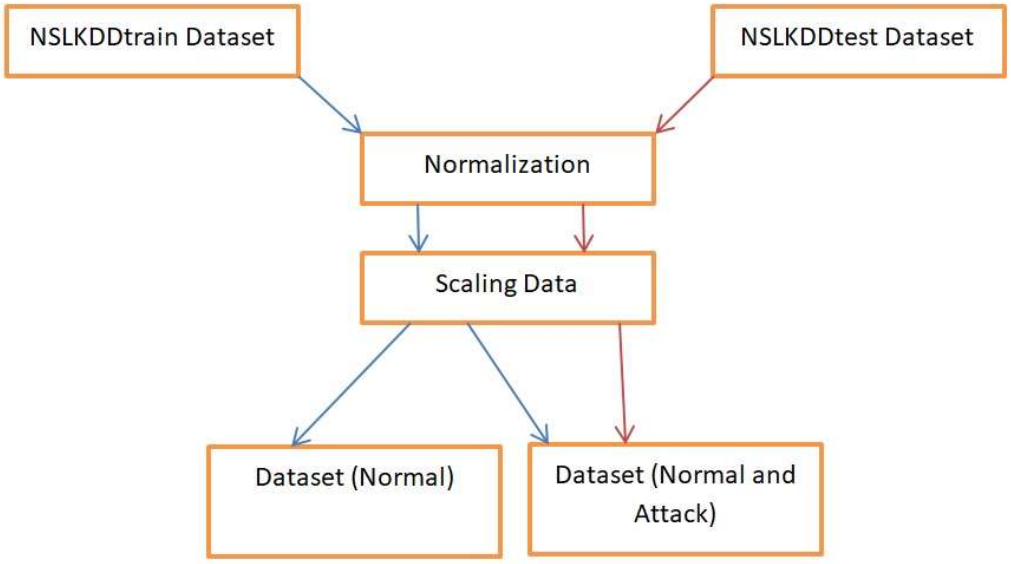

Figure 3.1: Data processing

When working on tuning models we need to consider some other datasets. This dataset helps to validate the model and is called validation dataset. The validation dataset is a sample of data from the available dataset which is not seen in the training data and gives us an estimate of the model's performance in terms of tuning the hyperparameters. The dataset is already split into training and testing sets. Then we split the training set further into an actual training set and a validation set. Out of the training set, 80% is used for training and 20% for validation.

3.4. Model Architecture

After the preprocessing and conversion of complete data, data modeling is conducted. In order to do this research we have selected Generative Adversarial Neural Network, Wasserstein Generative Adversarial Network and Deep Neural Network as a training and detection algorithms. We have got a number of advantages that are stated below.

- The complex internal structures make them to learn and accommodate large number of features and patterns.
- They have integral property of learning through interconnected artificial neurons during training.
- They can easily generalize form similar patterns through the knowledge they get it from training.
- As Generative adversarial neural network is generated new data from latent space and classify normal from attack as probability value.

To define model architecture of selected algorithms, we have taken in to two perspectives. The first one is setting default value for listed parameters to find out on each selected algorithms such as number of hidden layers, number of neurons on hidden layer, number of epochs (training iteration), activation function, loss function, learning rate, dropout, batch size and regularization which is based on keras framework.

The second one is to find out hyperparameters of each algorithm through tuning parameters which stated above to decide whether the training starts using default parameters or tuning hyperparameters is needed to find optimal set of hyperparameter values. The more hyperparameters of an algorithm needed to be tune, the slower the tuning process and computationally expensive. So we had to select a minimum subset of model hyperparameters to tune.

In Table 3.4 shown that hyperparameters to be selected for tuning and set the default values as taken with heuristic approach. In GAN, Discriminator network have an outer layer to have sigmoid activation function because of the neural network is classifier and sigmoid calculates the values to be between 0 and 1. Also the loss function is binary classification due to the output is normal or attack. For WGAN, the Discriminator network used Wasserstein distance for classification due to for stability of the training and classification purpose. On Generator side of

both GAN and WGAN the outer layer is normal Tanh activation function, because of the output is transferred to the Discriminator network as equal input size, however Generator network is not capable in classification purpose and tune through the Discriminator network. So only the Discriminator network will be optimized hyperparameter. On optimizer side Adam is used for gradient decent optimization due to its reliability and for models training to be more quickly [59].

Table 3.4: Hyperparameters selected for tuning and default values

Deep Learning Algorithms	GAN	WGAN
	Discriminator	Discriminator
Learning Rate	Selected	Selected
Number of hidden layers	Selected	Selected
Number of neurons on hidden layer	Selected	Selected
Batch size	Selected	Selected
Epoch	Selected	Selected
Activation function	LeakyRelu and Sigmoid	LeakyRelu and Sigmoid
Loss function	Binary cross entropy	Wasserstein distance
Optimizer	Adam	Adam

3.4.1. Grid Search

Grid search is the process of reading the data to configure optimal parameters for a given model which simply makes a complete search over a given subset of the hyperparameters space of the training algorithm [60]. Grid search is built a model on each parameter combination possible. It iterates through every parameter combination and stores a model for each combination. This phase helps to show that the parameters derived in the training phase work based on evaluation metrics. If the results are not satisfactory, then the model must be trained again to obtain better hyperparameter values which can produce accurate models. Both training and validation set are

passed to the grid search. Then the parameter grid (hyperparameters with default values) and algorithm are specified and passed to the cross-validation. The cross-validation run on the training data and searches the whole parameter grid for finding best parameters. The developed models are validated their effectiveness on the validation data. The selected optimal parameters are used to build the final model. The final model is then evaluated on the testing set.

3.4.2. Cross-validation

Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run and a dataset of unknown data (validation dataset) against which the model is tested (testing dataset). There are different types of cross-validation but in this research, a grid search with 5-fold cross-validation is used to evaluate and select the best subset of hyperparameters for the selected models. In 5-fold cross-validation, the original sample is randomly partitioned into 5 equal sized subsamples. Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 4 ($5 - 1$) subsamples are used as training data. To reduce variability, the cross-validation process is then repeated 5 times using different partitions.

Parameter space used in grid search as follow:

Learning Rate: 0.1, 0.01, 0.001 and 0.0001

Number of Hidden Layers: 1, 2 and 3 layers

Number of Neurons on Hidden Layer: 122 and 64

Batch Size: 128,500 and 1000

Epoch: 10,100 and 1000

Dropout: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9

3.5. Data Modeling

The dataset is noted to be of the form Xi, Ci, where Xi is the feature vectors of sample X (i = {1, 2... m}) and Ci is the class label of sample i (where Ci ∈ (normal, attack)). In order to solve the intrusion detection problem, Generative Adversarial Network-based Intrusion Detection System runs in two phases: training and testing as shown in Algorithm 3.1.

In the training phase, the system uses a training dataset and creates a model based on the proposed GAN and its variance WGAN model. At this stage, noise variables consisting of random numbers uniformly distributed in the (0, 1) range are input into G to generate examples of adversarial traffic. In other hand the X_{normal} data are entered in to Discriminator network from the training dataset. Then the system employs the model for identifying the label of testing data in the testing phase to estimate the performance of the model.

In the second phase, all data in the KDDTest+ dataset, including normal traffic transactions and attack traffic transactions, are pre-processed as training data did before, be de-labeled and digitized. These pre-processed data X_{test} will be sent to Discriminator network for anomaly detection in GAN and WGAN based IDS. The trained Discriminator can distinguish normal data from abnormal data and the output value of the range from 0 to 1. When the output value is 1, Discriminator determines that the input is normal traffic example through the prior adversarial learning in the training stage, otherwise the input is attack. However the Discriminator is tough to evaluate on the classification point to achieve Nash Equilibrium.

Algorithm 1:	GAN implementation on NIDS
Input :	KDDTrain+ → 1. Normal traffic transaction X_{normal}= {x_1,x_2,x_3…..x_m}from the training set; with m samples in defined batch size 2. The random variable noise *n*; KDDTest+ → 3. X_{test} which contains X_{normal} and X_{attack}
	Build Generator *G,* the Discriminator *D and GAN*; select mini-batch for training Xavier initialize Weight = 0 and Bias = 0
	for *Initialized GAN* **do**
	for *i* = 1*; training times* **do**
	Train Discriminator ; Train Generator
	for *G* **do**
	G generates the fake normal traffic examples X*generated* from *n* based on X_{normal}; *G* Passes X*generated* to *D*; Compute the loss function , optimizer update weight
	end
	for *D* **do**
	D classifies dataset including X*generated* and X*normal*;
	end
	end
	end
	for *Trained D* **do**
	D classifies the testing set KDDTest+, getting predicted labels; Ci
	end

Algorithm 3.1: GAN and implementation on NIDS

The WGAN algorithmic implementation is similar with GAN; however the difference is stated as follows:

- Use a linear activation function in the output layer of the critic model (instead of sigmoid).
- Use Wasserstein loss to train the critic and Generator models that promote larger difference between scores for real and generated images.
- Constrain critic (Discriminator) model weights to a limited range after each mini batch update (-c, c) [38] .

In Figure 3.2 X_{normal} data enter to Discriminator Network to classify as real or normal. Also Noise n is entered to Generator Network to generate fake or attack data, then in order to fool transfer to the Discriminator and classified as generated or attack malicious intrusion. The Cost function is tuning both Generator and Discriminator by update weight and bias to continuously to play a min-max game where one is trying to outsmart the other.

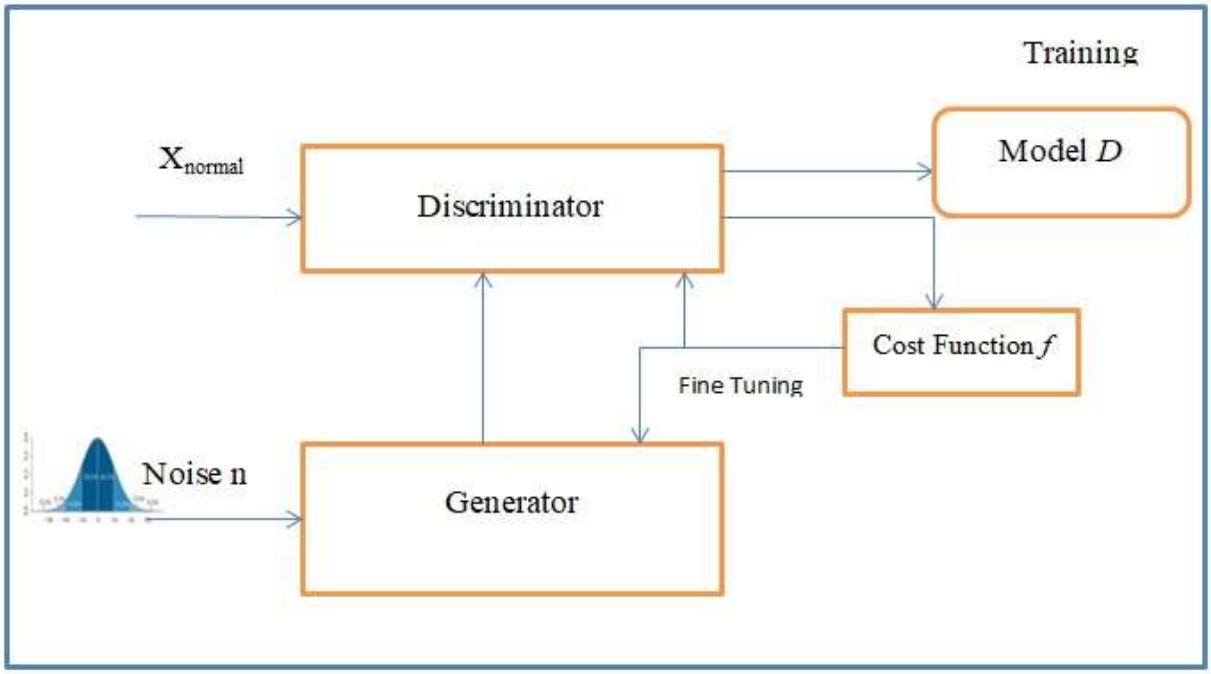

Figure 3.2: Baseline training

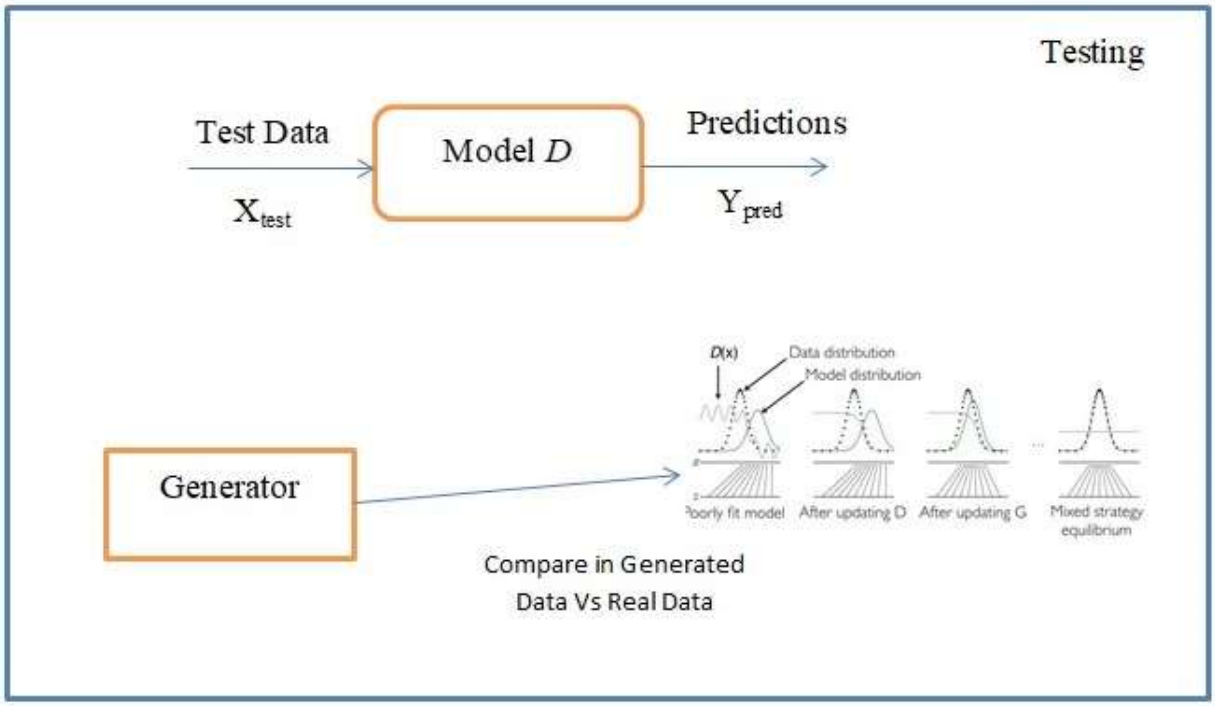

Figure 3.3: Testing

In Figure 3.3 shown testing of the model and predict the output from the training phase and Generator try to close generated data to the original distribution of data.

3.6. Evaluation Metrics

There are many different methods to measure the performance of GAN module prediction. The researcher [61] reviewed and critically stated more than 24 quantitative and 5 qualitative measures for evaluating generative models with a particular emphasis on GAN-derived models. Most of the measurements are mainly concerned on measuring Image classification and generation. However in the case of evaluate the performance of the GAN based IDS models is different and challenging in classification because of the output on Discriminator classify in the probability of normality or abnormality.

The most common performance measures in the field of IDS deep learning are precision, recall, and F1 score. In these metrics for evaluating IDS, the higher the value represents that the model performs better. But in some cases the precision and recall are contradictory, so the evaluation can only considered the balance according to the requirements of task. Thus, F1 score becomes an important indicator, the higher the score, the better performance that the IDS has.

Since the experiments in this work is mainly to discriminate normal records and malicious attack records, a confusion matrix of 2-classes classifiers is adopted to calculate the performance metrics and the comparison between GAN and WGAN. Based on Ali Borji's studied from 29 metrics stated, we have chosen precision, recall and F1 score based on high detecting over fitting, disentangled latent space and well-defined boundary (0,1) [61]. Lucic proposed to compute precision, recall and F1 score to quantify the degree of over fitting in GANs [53].

The confusion matrix is a table that describes the classification results in detail. Each column of the matrix represents the instance in the prediction class and each row represents the instance in the actual class. The results can be summarized into the following four basic situations:

- True Positive (TP): Normal records are correctly discriminated by the model.
- False Negative (FN): Malicious attacks are incorrectly identified as the normal records.
- False Positive (FP): Normal records are incorrectly discriminated to be anomaly.
- True Negative (TN): Malicious attacks are successfully identified by the model

Table 3.5: Confusion Matrix

		Prediction	
		Normal	Attack
Actual	Normal (Negative)	True Negative (TN)	False Positive (FP)
	Attack (positive)	False Negative (FN)	True Positive (TP)

From these cases of confusion matrix, classification indicators such as [4]: Accuracy, Precision, Recall (Sensitivity), F1 score and AUC (Area under the Receiver Operating Characteristic (ROC) Curve) can be further calculated.

Accuracy: is defined as the ratio of correctly classified samples to total samples. Accuracy is a suitable metric when the dataset is balanced. In real network environments; however, normal samples are far more abundant than are abnormal samples; thus, accuracy may not be a suitable metric. The accuracy of the proposed system is calculated using Equation 3.1

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.1)$$

Precision: is defined as the ratio of true positive samples to predicted positive samples; it represents the confidence of attack detection.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.2)$$

Recall: is defined as the ratio of true positive samples to total positive samples and is also called the detection rate. The detection rate reflects the model's ability to recognize attacks, which is an important metric in IDS.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.3)$$

F1-score: is defined as the harmonic average of the precision and the recall.

$$\text{F1-score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \quad (3.4)$$

The false positive rate (FPR): is defined as the ratio of false positive samples to predicted positive samples. In attack detection, the FPR is also called the false alarm rate.

$$\text{FPR} = \frac{FP}{TN + FP} \quad (3.5)$$

The purpose of IDS is to recognize attacks; therefore, attack samples are usually regarded as positives, and normal samples are usually regarded as negatives. In attack detection, the frequently used metrics include accuracy, recall (or detection rate), FNR (or missed alarm rate), and FPR (or false alarm rate).

CHAPTER FOUR

EXPERIMENT

4.1. Overview

In this Chapter the implementation detail for anomaly-based intrusion detection experiment presented. The specific model refers to the existing code provided by many previous works, and the main part is completed by making appropriate modifications according to the code published by Erik Linder Noren on GitHub [62]. We have used this implementation to evaluate the performance of the proposed work which is the GAN and WGAN anomaly detection and the evaluation presented here. In this Chapter we will cover overview of how the implementation is done, tools used to do this experiment, how components of the system are implemented and finally measure the performance of the work done.

4.2. Experimental Setup

4.2.1.Tools Used

According to the GAN and WGAN code provided by Erik on GitHub platform based on Keras framework for artificial image generation, in this experiment, the code is modified corresponding to IDS are implemented. In terms of the hardware environment, the test was performed on a laptop equipped with an Intel(R) core I5-7900 CPU (3.30GHz), 4GB RAM and a Linux Ubuntu 20.04 operating system. The experimental simulations were performed using Google Colab research framework with TensorFlow2.0 as back end on Keras and Scikit-learn on Juypter notebook. These are the most commonly used machine learning frameworks, and Python 3 was used as the programming language which are available as open source.

4.2.1.1. TensorFlow2.0

Tensor flow is Google's open source deep learning library released on November 2015. It includes C++ and Python API's. It has plenty of abstraction power but users might also be working with computational primitive wrappers such as matrix operations, element-wise math operators, and looping control. Tensor flow considers networks as a directed graph of nodes with

data flow computation and dependencies encapsulated in it. Tensor flow is used as a back end for Keras library which is used for development of deep neural network classifiers.

4.2.1.2. Keras

Keras is a Python machine learning library for deep learning that can run using Theano or Tensor Flow as a back end. It is main focus is enabling implementation of deep learning models as fast and easy as possible for research and development. In this experiment, the code is implemented using Python 3 and can execute on GPUs and CPUs given the underlying frameworks. Keras is used for the development of deep neural network classifiers for proposed GAN and WGAN IDS model.

4.2.1.3. Scikit Learn

Scikit-learn is a Python machine learning library which includes a wide range of state-of-the-art machine learning algorithms for supervised and unsupervised problems. It focuses on bringing machine learning to non-specialists by providing a general-purpose high-level language. It is easy to use, has high performance and contains detailed. Scikit learn is used to evaluate Discriminator network to classify by calculating performance metrics.

4.2.1.4.Activation Function

An activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data**.** When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron**.** Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1 [63].

A. ***Leaky ReLU*****:** allows the pass of a small gradient signal for negative values. As a result, it makes the gradients from the Discriminator flows stronger into the Generator. Instead of passing a gradient (slope) of 0 in the back-prop pass, it passes a small negative gradient. In this experiment we have used in every layer except output layer of Discriminator and Generator network.

B. ***Sigmoid*****:** Just like any other binary classification model, the output of the Discriminator is a single number between 0 and 1, which can be interpreted as the probability of the input data being fake i.e. generated. This method is generally used for binary

classification problems. So we have used it at output layer of Discriminator network to classify normal and attack.

C. ***Tanh*:** It is very similar to the sigmoid except that the output values are in the range of -1 to +1. Thus, tanh is said to be zero centered. The difference between the sigmoid and tanh is that the gradients are not restricted to move in one direction for tanh. Thus, tanh is likely to converge faster than the sigmoid function.it is implemented on last output layer of Generator in order to increase convergence and the output also is not binary.

4.2.1.5. Optimizer

Adam: Adaptive Moment Estimation is a popular version of gradient descent because it automatically tunes itself and gives good results in a wide range of problems. Adam is used as an optimizer to speed up the convergence, and the weight decay is set to 0.005 to prevent over fitting. It works better faster and more reliably reaching a global minimum when minimizing the cost function in training [59].

4.2.2. Implementation of the Components

In our experiment, we constructed a Keras deep neural network with an input layer, three hidden layers and an output layer as described in Figure 4.1. On the Discriminator Network the input dimension is 122 and the output dimension is 1. Whereas in Generator network the input noise dimension are 100 and the output dimension are 122. The hidden layers contain 122 on both Generator and Discriminator neural network at each 3 hidden layers with dropout rate is 0.6 for GAN and 0.5 for WGAN. Our model initiation parameters are for the batch size 1000, for the epoch reached 1000 epochs. The learning rate is 0.001 for GAN and 0.0001 for WGAN. The loss function is used binary cross entropy for GAN and Wasserstein distance for WGAN. It is independent for each vector component (class), meaning that the loss computed for every Discriminator output vector component is not affected by other component values. According to our objective is to identify anomaly intrusion detection, it is used for 2-class classification, were the insight of an element belonging to a certain class either normal or attack.

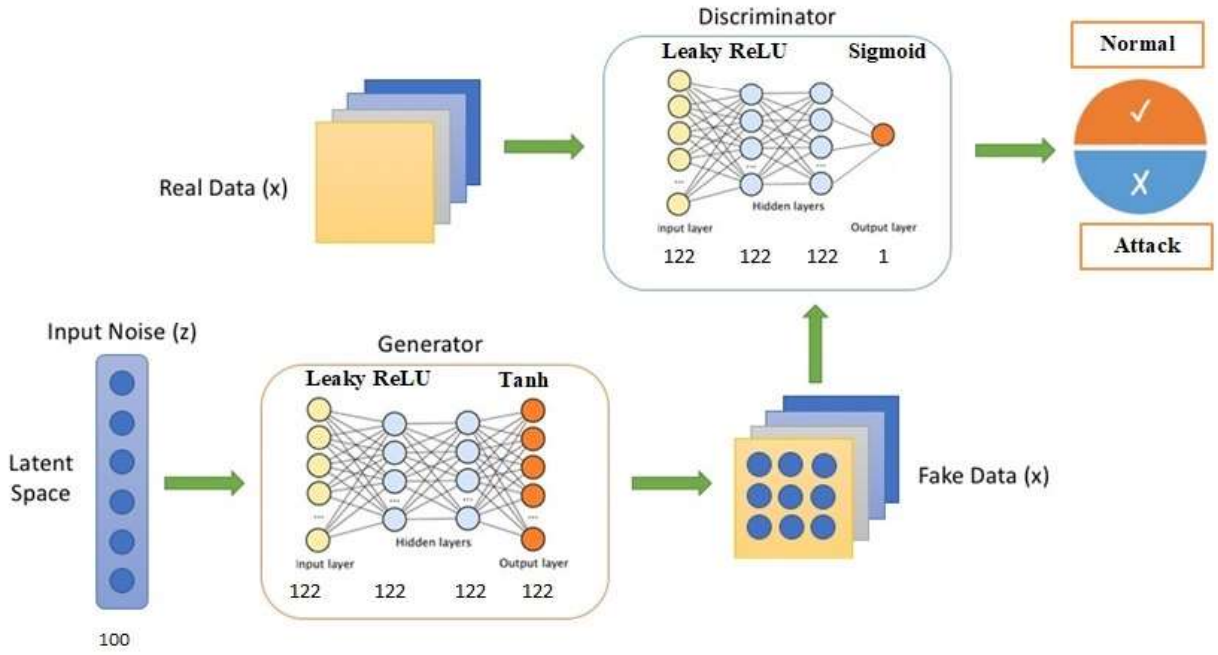

Figure 4.1: Implementation Components on GAN and WGAN

4.2. Experimental Scenarios

To evaluate our hypothesis different experiments are conducted. Generally these experiments can be grouped in to two. The first part of experiments are aimed at evaluate the performance of GAN and WGAN model with default parameter. The second part of experiments is to evaluate the performance of GAN and WGAN with tuned hyperparameter and comparison with DNN model.

The results might be varying given the stochastic nature of the learning algorithm. Nevertheless, the general structure of training should be very similar. First, the loss and accuracy of the Discriminator and loss for the Generator model are reported to the console each iteration of the training loop. This is important. A stable GAN will have a Discriminator loss around 0.5, typically between 0.5 and maybe as high as 0.7 or 0.8. The Generator loss is typically higher and may hover around between 1.0 and 2.0 or even higher. The accuracy of the Discriminator on both real and generated (fake) data will not be 50%, but should typically hover around 70% to 80%. For both the Discriminator and Generator, behaviors are likely to start off unreliable and move around a lot before the model converges to a stable equilibrium [64].

CHAPTER FIVE

ANALYSIS AND RESULT

For the scope of this paper, the NSL KDD pre-processed dataset was fed into Deep Learning algorithms that are GAN and WGAN in order to compare with the paper is done on DNN [47]. This chapter will present all experimental results and corresponding analysis results.

5.1. Analysis of Grid Search Result

Table 5.1 shows that the result of grid search optimal hyperparameter and default parameter values that pass through grid search using keras default initializer framework.

Table 5.1: Optimal hyperparameter and default parameters of deep learning algorithms

Deep Learning Algorithms	GAN Discriminator		WGAN Discriminator	
	Default value	Hyperparameter	Default value	Hyperparameter
Learning Rate	0.001	0.001	0.001	0.0001
Number of hidden layers	1	3	1	3
Number of neurons on hidden layer	122	122	122	122
Batch size	32	1000	32	1000
Epoch	100	1000	100	1000
Dropout	0.5	0.6	0.5	0.5

5.2. Performance Evaluation

In this Section, the experimental results on both scenarios learning with default parameters and learning using best parameters by applying the hyperparameter technique for the selected classification algorithms are interpreted. All the classification models are evaluated by using a confusion matrix. This evaluation metrics shows the number of correct and incorrect predictions made by the classification model compared to the target values in the data by calculating the

number of test samples using four categories of evaluation metrics. This was previously stated in Chapter 3 under 3.6 evaluation metrics. Using default parameters, the WGAN has obtained 89 % of accuracy, 77.6% of recall, 78.1% of precision and 77.8% of F1-score. While the accuracy 47.8%, precision 45.2%, recall 99.9% and F1 score 62.3% found for the GAN as shown in Tables 5.2. The WGAN classifier is the best performer in case of accuracy and compared to the GAN.

Table 5.2 : Model classifiers with default parameters

Type of Algorithm	Accuracy	Recall	Precision	F1 Score
GAN	47.8%	99.9%	45.2%	62.2%
WGAN	89%	77.6%	78.1%	77.8%

To improve the accuracy of the two algorithms further, a hyperparameter tuning method is performed. Table 5.2, shows the accuracy, precision, recall, and f1-score results of the two algorithms before the hyperparameter tuning technique is applied. As can be seen in Tables 5.2 and 5.3, the accuracy of the two classify algorithms trained on their default parameters is lower than the accuracy gained by trained on best parameters by using the hyperparameter tuning technique. WGAN has good performance in case of accuracy than GAN after optimizing the hyperparameters indicates as a better classier in all metrics.

Table 5.3: Model classifiers with after hyperparameter tuned

Type of Algorithm	Accuracy	Recall	Precision	F1 Score
GAN	93.4%	93%	96.6%	94.8%
WGAN	95.7%	96.9%	96.9%	96.8%

5.3. Comparison of Classification Models

To compare with the existing anomaly detection techniques in references [47] [65], we conduct experiments on NSL-KDD dataset. In reference [47], network anomaly detection based on NSL-KDD dataset was explored. The research used the learning rate as its constant at 0.01 while the other two parameters number of neurons and number of hidden layers where optimized. The count of the neurons in a layer was experimented by changing it over the range of 2 to 1024. And the number of hidden layer in the network used 1 up to 5 layers. Also the dropout

is to regularize each layer at 0.01. The detection accuracy of the algorithms was found at DNN using 3 hidden layers and score 93% of accuracy, 99.7% of precision, 91.5% of recall and 95.5% of f1 score.

In reference [65], the author proposed a supervised based feature selection method based on data mining algorithms to identify relevant features from the NSL-KDD dataset. The detection accuracy of the algorithms Wrapper-Bayesnet feature selection method and classification was compared to Filter-Bayesnet, Wrapper Naïve Bayes, Filter Naïve Bayes and Wrapper-Bayesnet method was obtained an accuracy of 95.3%.

The optimization technique used in our proposed work has ease of calculation than the other methods used in the two referenced papers. The detection accuracy of the studies is also lower than our approach which WGAN scored accuracy of 95.7%. Based on these comparison results, our approach outperforms the studies in references [47] [65] in network anomaly detection.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1. Conclusion

Nowadays a growing of interconnected devices and services lead a world communication environment more complex and undetermined by human capability. Computer networks are dynamic, growing, and continually evolving with assisting human communication and integration of systems and services. Hackers or intruders have been affecting this interconnected environment by disrupting or break up with steal of information for personal purpose or advance. As complexity grows, it becomes harder to effectively communicate to human decision-makers the results of methods and metrics for monitoring networks, classifying traffic, and identifying malicious or abnormal events. Security experts require tools that support them understand the reason for, and make decisions about the information their analytic systems produce.

In order to support security experts, in this data driven world using deep learning algorithms as back-end engine is more support automatically to identify malicious and normal network traffics. In this paper we have proposed using Generative Adversarial Networks undoubtedly due to the fact that this kind of algorithm is relatively new and even though generate data for imbalanced class with in dataset. This work has shown the effect of parameters and neural network structure on the performance of GAN based IDS. The experimental results have shown that number of hidden layers, dropouts, batch size and epoch (iteration of model training) of the Discriminator model have significant effects on IDS performance for learning pattern in the dataset.

6.2. Future Work

Our work has shown a relatively good result in detecting attacks however it is necessary to improve our model further to detect more known and unknown attacks.

In addition, a further work that could be an extension of our work to fulfill the need as follows:

- Using different hyperparameter optimization technique to improve and identify core difference parameters that influence the model performance.

- Study on additional features and dataset included and selecting relatively high performance models.
- Hybrid IDS have shown high performance in other studies. So integrating with other signature-based IDS to form a hybrid IDS and measure the performance to what extent is usable the model.
- Implement with front end applications and using a model for analysis as a back end engine on live network traffic and measure the effectiveness of the whole system.

Reference

[1] A. Ometov, S. Bezzateev and J. Kannisto, "Facilitating the Delegation of Use for Private Devices in the Era of the Internet of Wearable Things," *IEEE Internet of Things Journal,* vol. 4, no. 4, pp. 843-854, August 2017.

[2] W. Spencer, "tech-faq.com," 05 April 2019. [Online]. Available: http://www.tech-faq.com/network-attacks.html. [Accessed 10 05 2020].

[3] A.-M. Research, "Kaspersky Security Bulletin 2019 Statistics," Kaspersky, 2019.

[4] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," *Applied Sciences,* vol. 9, no. 20, pp. 43-96, 2019.

[5] S. Mohammadi, M. H. Sherkat and M. Jamporazmey, "A taxonomy framework based on ITU-TX-805 security architecture for quantitative determination of computer network vulnerabilities," *Security And Communication Networks,* vol. 6, p. 864–880, 2013.

[6] S. T. F. Al-Janabi and H. A. Saeed, "A Neural Network Based Anomaly Intrusion Detection System," *Developments in E-Systems Engineering,* 2011.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems,* vol. 3, pp. 2672-2680, 2014.

[8] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzama, "Survey of intrusion detection systems: techniques, datasets and challenges," 2019.

[9] Q. Niyaz, W. Sun, A. Y Javaid and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *9th EAI International Conference on Bio-inspired Information and Communications Technologies*, Toledo, 2016.

[10] D. Mwiti, 02 07 2018. [Online]. Available: https://heartbeat.fritz.ai/introduction-to-generative-adversarial-networks-gans-35ef44f21193. [Accessed 15 12 2020].

[11] A. A.Ghorbani, W. Lu and M. Tavallaee, Network Intrusion Detection and Prevention: Concepts and Techniques, 1st ed., New York: Springer Science & Business Media, 2019.

[12] A. Adeyemo , "Design of an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS) for the EIU Cybersecurity Laboratory," 2016.

[13] N. Sainis, D. Srivastava and R. Singh, "Classification of various Dataset for Intrusion Detection System," *International Journal of Emerging Technology and Advanced Engineering,* vol. 8, no. 1, 2018.

[14] M. Kebede , "K -Means Clustering and Random Forest Based Hybrid Intrusion Detection Algorithm," 2017.

[15] J. Sen and S. Mehtab, "Machine Learning Applications in Misuse and Anomaly Detection," 2019.

[16] K. Letou , D. Devi and Y. J. Singh , "Host-based Intrusion Detection and Prevention System (HIDPS)," *International Journal of Computer Applications ,* 2013.

[17] A. Youssef and A. Emam, "Network Intrusion Detection Using Data Mining and Network Behaviour Analysis," 2011.

[18] D. Li, D. Chen, J. Goh and S.-K. Ng, "Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series," 2019.

[19] P. García-Teodoro, J. . E. Díaz-Verdejo, G. Maciá-Fernández and E. Vázquez, "Anomaly-based network intrusion detection:Techniques, systems and challenges," *Computers & Security,,* vol. 28, no. 1, pp. 18-28, 2009.

[20] J. Lee, S. Moskovics and L. Silacci, "A Survey of Intrusion Detection Analysis Methods," 1999.

[21] P. Mishra, E. S. Pilli and V. Varadharajan, "Securing Virtual Machines from Anomalies Using ProgramBehavior Analysis in Cloud Environment," in *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems*, 2016.

[22] S. Naseer and Y. Saleem, "Enhanced Network Intrusion Detection using Deep Convolutional Neural Networks," *KSII Transactions on Internet and Information Systems,* vol. 12, no. 10, pp. 5159-5178, 2018.

[23] L. Mohammadpour, T. C. Ling, C. S. Liew and C. Y. Chong, "A Convolutional Neural Network for Network Intrusion Detection System," in *THE 46TH MEETING OF THE ASIA-PACIFIC ADVANCED NETWORK*, Auckland, 2018.

[24] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access,* vol. 5, p. 21954–21961, 2017.

[25] M. Alom, V. R. Bontupalli and T. M. Taha, "Intrusion Detection using Deep Belief Networks," in *National Aerospace and Electronics Conference (NAECON)*, 2015.

[26] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim and K. J. Kim, "A survey of deep learning-based network anomaly detection," 2017.

[27] F. Farahnakian and J. Heikkonen, "A Deep Auto-Encoder based Approach for Intrusion Detection System," in *20th International Conference on Advanced Communications Technology(ICACT)*, 2018.

[28] A. Gouveia and M. Correia, "A Systematic Approach for the Application of Restricted Boltzmann Machines in Network Intrusion Detection," in *14th International Work-Conference on Artificial Neural Networks*, Cadiz, 2017.

[29] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*, Berlin, Springer, 2012, pp. 599-619.

[30] T. Merino, M. Stillwell, M. Steele, M. Coplan, J. Patton, A. Stoyanov and L. Deng, "Expansion of Cyber Attack Data from Unbalanced Datasets Using Generative Adversarial Networks," in *Software Engineering Research Management and Applications (SERA) Studies in Computational Intelligence*, 2019.

[31] M. Shahriar, N. I. Haque, M. A. Rahman and M. Alonso Jr, "G-IDS: Generative Adversarial Networks Assisted Intrusion Detection System," in *2020 IEEE 44th Annual Computers, Software and Applications*, 2020.

[32] J. Brownlee, "A Gentle Introduction to Generative Adversarial Networks," 17 June 2019. [Online]. Available: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/. [Accessed 13 10 2020].

[33] C. Wang, C. Xu, X. Yao and D. Tao, "Evolutionary Generative Adversarial Networks," 2019.

[34] A. Gharakhanian, "generative-adversarial-networks-hot-topic-machine-learning," 3 January 2017. [Online]. Available: https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html. [Accessed 13 10 2020].

[35] L. Weng, "From GAN to WGAN," 2019.

[36] X. Mao, Q. Liy, H. Xiez, . R. Y.K. Laux, Z. Wang and S. P. Smolleyk, "Least Squares Generative Adversarial Networks".

[37] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[38] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein GAN," in *the 34th International Conference on Machine Learning (ICML)*, 2017.

[39] M. Zamani, "Machine Learning Techniques for Intrusion Detection," 2013.

[40] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adverserial Networks," 2017.

[41] V. Nagarajan and J. Kolter, "Gradient descent gan optimization," in *Advances in Neural Information Processing Systems*, 2017.

[42] D. Nautiyal, "Underfitting and Overfitting in Machine Learning," 18 May 2020. [Online]. Available: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. [Accessed 18 11 2020].

[43] X.-Y. C. H. Z. L.-D. X. H. L. S.-H. D. Jia Wu, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," *Journal of Electronic Science and Technology,* vol. 17, no. 1, pp. 26-40, 2019.

[44] jeremyjordan.me, "Hyperparameter tuning for machine learning models," 2 November 2017. [Online]. Available: https://www.jeremyjordan.me/hyperparameter-tuning/. [Accessed 18 10 2020].

[45] K. Han, S. Naseer, Y. Saleem, S. Khalid and J. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Network," *IEEE Access,* vol. 6, pp. 1-1, 2018.

[46] M. Rigaki, "Adversarial Deep Learning Against Intrusion Detection Classifiers," Luleå, 2017.

[47] R. Vigneswaran, V. R and S. Kp, "Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security," *Conference: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT),* pp. 1-6, 2018.

[48] H. Zhang, X. Yu, P. Ren, C. Luo and G. Min, "Deep Adversarial Learning in Intrusion Detection: A Data Augmentation Enhanced Framework," 2019.

[49] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein and J. D. Tygar, "Adversarial machine learning," in *the 4th ACM workshop on Security and artificial*, 2011.

[50] K. Grosse, N. Papernot, P. Manoharan, M. Backes and P. McDaniel, "Adversarial Perturbations Against Deep Neural Networks for Malware Classification," *arXiv preprint,* 2016.

[51] O. M. EZEME, Q. H. MAHMOUD and A. AZIM, "Design and Development of AD-CGAN:Conditional Generative Adversarial Networks for Anomaly Detection," vol. 8, p. 177667–177681, July 2020.

[52] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani and D. Vitali, "Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers,"

International Journal of Security and Networks, vol. 10, no. 3, pp. 137-150, 2015.

[53] M. Lucic, K. Kurach, M. Michalski, O. Bousquet and S. Gelly, "Are gans created equal? a large-scale study," 2018.

[54] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA),* 2009.

[55] "NSL-KDD dataset," 2009. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html. [Accessed 17 09 2020].

[56] L. D. a. D. 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,* vol. 4, no. 6, 2015.

[57] . J. J. Davis and A. J. Clark, "Data Preprocessing for Anomaly Based Network Intrusion Detection : A review," *Computer & Security,* vol. 30, pp. 353-375, 2011.

[58] D. Srivastava, "Classification of various Dataset for Intrusion Detection System," 2018.

[59] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017.

[60] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for Neural Architecture Search," 2019.

[61] A. Borji, "Pros and Cons of GAN Evaluation Measures," 2018.

[62] E. L. Norén, "Keras-GAN," 2018. [Online]. Available: https://github.com/eriklindernoren/Keras-GAN. [Accessed 18 09 2020].

[63] V. Jain, 30 December 2019. [Online]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253. [Accessed 13 11 2020].

[64] J. Brownlee, "How to Identify and Diagnose GAN Failure Modes," 8 July 2019. [Online]. Available: https://machinelearningmastery.com/practical-guide-to-gan-failure-modes/. [Accessed 13 10 2020].

[65] M. Teffera, "A data mining approach for Intrusion Detection System Using Wrapper Based Feature Selection Method," *Masters Thesis,* June 2014.

[66] M. Rigaki and S. García, "Bringing a GAN to a Knife-Fight: Adapting Malware Communication to

Avoid Detection," in *2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, 2018.

APPENDICES

Appendix A: Hyperparameter Optimization using Grid Search from Keras Framework

```
[ ] ######################################## Finding Optimal Grid Parameters for a Discriminator model ##############################################
    ###### Function to linearly decrease or increase the number of nodes for the layers between the first layer and last layer ######################
    def FindLayerNodesLinear(n_layers, first_layer_nodes, last_layer_nodes):
        layers = []

        nodes_increment = (last_layer_nodes - first_layer_nodes)/ (n_layers-1)
        nodes = first_layer_nodes
        for i in range(1, n_layers+1):
            layers.append(math.ceil(nodes))
            nodes = nodes + nodes_increment

        return layers

    ####### Function to vary the parameters of a tensor flow model by creating a new model based on given parameters ##############################
    def createmodel(n_layers, first_layer_nodes, last_layer_nodes, activation_func):
        model = Sequential()
        n_nodes = FindLayerNodesLinear(n_layers, first_layer_nodes, last_layer_nodes)
        for i in range(1, n_layers):
            if i==1:
                model.add(Dense(first_layer_nodes, input_dim=X_train.shape[1], activation=activation_func))
            else:
                model.add(Dense(n_nodes[i-1], activation=activation_func))

        #Finally, the output layer should have a single node in binary classification
        model.add(Dense(1, activation='sigmoid'))
        model.compile(optimizer='adam', loss= 'binary_crossentropy', metrics = ["accuracy"]) #note: metrics could also be 'mse'

        return model
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=<tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7f67c266fbe0>,
             iid='deprecated', n_jobs=None,
             param_grid={'activation_func': ['relu', 'tanh'],
                         'batch_size': [1000], 'epochs': [100],
                         'first_layer_nodes': [122, 64],
                         'last_layer_nodes': [122, 64], 'n_layers': [1, 2, 3]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
################################  Optimal Grid Parameters number of nurons and number of layers ##################################################################
print(grid.best_score_)
print(grid.best_params_)
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
pd.DataFrame(grid.cv_results_)[['mean_test_score', 'std_test_score', 'params']].to_csv('gdrive/My Drive/Colab Notebooks/GAN Hyperparameter/GridOptimizatio
```

```
0.9948346376419067
{'activation_func': 'relu', 'batch_size': 1000, 'epochs': 100, 'first_layer_nodes': 122, 'last_layer_nodes': 122, 'n_layers': 3}
0.971675 (0.002782) with: {'neurons': 64}
0.971252 (0.002945) with: {'neurons': 122}
```

Appendix B: Discriminator and Generator Model on GAN

```
 #random_normal
 #glorot_uniform
# define the standalone discriminator model
def define_discriminator(data_shape= 122):
  model = Sequential()
  model.add(Dense(122, input_dim=data_shape, kernel_initializer='random_normal', activation='relu'))
  model.add(Dropout(0.6))
  model.add(Dense(122, input_dim=122, kernel_initializer='random_normal', activation='relu'))
  model.add(Dropout(0.6))
  model.add(Dense(1, kernel_initializer='random_normal', activation='sigmoid'))
  # compile model
  opt = Adam(lr=0.001, beta_1=0.6)
  model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
  return model

# define the standalone generator model
def define_generator(latent_dim):
  model = Sequential()
  model.add(Dense(122, input_dim=latent_dim, kernel_initializer='glorot_uniform', activation='relu'))
  model.add(Dropout(0.6))
  model.add(Dense(122, input_dim=122, kernel_initializer='glorot_uniform', activation='relu'))
  model.add(Dropout(0.6))
  model.add(Dense(122, kernel_initializer='glorot_uniform', activation='tanh'))
  return model
```