

**Network Intrusion Detection
Using Generative Adversarial
Networks**

Master Thesis

Author: Xiran Zhang

Supervisor: Dr. Dongseong Kim

A thesis submitted in fulfilment of the requirements for the
degree of Masters of Computer Science

in

Computer Science and Software Engineering

University of Canterbury

Network Intrusion Detection Using Generative Adversarial Networks

College of Science
Computer Science and Software Engineering
Masters of Computer Science

Xiran Zhang

Abstract

Intrusion detection systems (IDS), as one of important security solutions, are used to detect network attacks. With the extensive applications of traditional machine learning algorithms in the security field, intrusion detection methods based on the machine learning techniques have been developed rapidly. However, since the progress of technology and the defects of the intrusion detection system based on machine learning algorithms, the system has gradually failed to meet the requirement for cyber security. Generative Adversarial Networks (GANs) have been widely studied and applied in anomaly detection in recent years thanks to their high potential in learning complex high-dimensional real data distribution. Deep learning techniques can greatly overcome the disadvantages of using traditional machine learning algorithms for intrusion detection. This work proposes to use current existing GANs and their variants for network intrusion detection using real dataset and show the feasibility and comparison results.

Keywords: cybersecurity, intrusion detection, intrusion detection system, generative adversarial networks.

Contents

1	Introduction	4
1.1	Motivation for Study	5
1.2	Aim and Scope	6
1.3	Research Questions	7
1.4	Organisation of the Thesis	7
2	Background	8
2.1	Deep Learning in intrusion detection	11
2.2	Generative Adversarial Networks	14
2.3	Bidirectional Generative Adversarial Networks	16
3	Proposed Methods	18
3.1	Data Pre-processing	19
3.2	Experiments Models	20
3.3	Evaluation Methods	23
3.4	Evaluation Metrics	24

4	Results and Analysis	26
4.1	Performance of GAN Based IDS in Different Value of Learning Rate .	26
4.2	Performance of IDSes with Different Amounts of Batch Sizes and Epochs	27
4.3	Performance of IDSes with Different Number of Layers of Neural Network Hidden Layers	30
4.4	Limitations of Study	33
4.5	Future Work	34
5	Conclusions	35

Chapter 1

Introduction

With the development and popularization of network and information technology, the digital world has been deeply and extensively integrated into every aspect of daily social activities. Individuals, organizations and governments are increasingly keeping important and confidential data on the Internet. With the growth requirement of internet, information security field is facing more and more serious challenges from more and more diversified threats. Intrusion detection as an active defense technology is one of the most important unavoidable solution to use in cybersecurity area. Intrusion detection systems (IDS) use the distribution of sample data to build an intrusion detection model to identify unauthorized activities on system and computer networks that may pose threats to information confidentiality, integrity, or availability (CIA) [1]. An IDS issues intrusion alerts when such attacks are detected to respond to the attacks. The main goal of an IDS is to classify network traffic and computer usage between malicious ones and normal ones which traditional stateless firewalls cannot perform.

In the past decades, machine learning algorithms have been widely used in the improvement on intrusion detection systems due to their high efficiency, flexibility and deployability. Nowadays, IDSes based on machine learning techniques have become the mainstream. However, due to the huge quantification and complexity of malicious attacks, some shortcomings of traditional machine learning algorithms have been amplified, such as the emphasis on processing low-dimensional data and

the lack of response to high-dimensional data, and the reliance on manual features selection.

The manual feature selection process omitted by deep learning has become a practical solution for machine learning tasks that process high-dimensional data. In recent years, the application of deep learning algorithms in the field of intrusion detection has developed rapidly. Algorithms such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and Automatic Encoder (AE) have further improved the accuracy and simplicity of IDSes [2–5].

Generative adversarial networks (GAN) are a class of deep learning algorithms that implemented by two neural networks contesting with each other in a two-player game framework. Since Goodfellow et al. [6] first published GAN’s paper in 2014, GAN have shown their advanced advantage in generating higher-dimensional data such as images, sounds asnd text. In addition, the improvement of anomaly detection performance in other fields, especially medical imaging in the field of image, has made outstanding contributions [7–9]. GAN has the following advantages in the field of intrusion detection. First, GAN has a great potential to learn to mimic any data distribution, allowing GAN to generate real data so an IDS can benefit from more available data. Second, compared with other deep learning algorithms, GAN has inherent potential and advantages when faced with adversarial attacks, such as attackers generate malicious traffic and makes it similar to normal traffic in the hope of tricking IDS into classifying it in the wrong class. Recurrently, there are many studies using GAN to improve IDS or develop new attack patterns such as generate adversarial malware examples [10–13]. But researches on GAN used in intrusion detection are still scarce.

1.1 Motivation for Study

This paper will explore the influence of various parameters of GAN and its variant Bidirectional Generative Adversarial Networks (BiGAN) on its detection efficiency in intrusion detection. As more and more information security researchers pay at-

tention to GAN's application in intrusion detection, many excellent cases of GAN and IDS are put forward. But these cases tend to focus on transverse comparisons with IDS based on traditional machine learning algorithms and ignore their own vertical comparisons.

There are many decisive factors, the first issue is how many the number of hidden layers of the neural network is used. According to the hidden layer theory of neural network, in general, the more layers, the better the training effect will be. An over-fitted intrusion detection model will seriously reduce the prediction effect of the model on the test data [14]. The second issue is training times. When a complete dataset passes through the neural network once and returns once, this process is called an epoch. As the number of epochs increases, the number of weight updates in the neural network also become increased, and the curve changed from under-fitting to over-fitting, while the appropriate epochs had no accurate answer, and the answers were different for different models and datasets. The third issue is about the number of samples; One of the prominent characteristics of GAN algorithms is to generate the fake data according to the existing real data. The batch size of noise and real data as inputs has an indispensable influence on the quality of the generated data.

Among the GAN variants, BiGAN is a promising one. It combines the autoencoder structure into a generic GAN framework, which is a typical variation of GAN in structure [15]. It corrected for GAN's difficulty in guessing the properties of one pixel based on another by learning the data distribution in latent space. Some previous studies have shown that its stability is better than traditional machine learning algorithms and naive GAN algorithm.

1.2 Aim and Scope

The purpose of this thesis is to evaluate models in terms of performance by running IDS based on GAN and BiGAN with different parameters. The research scope includes background research, experiment implementation and testing, and perfor-

mance index collection and analysis. This study will start with the collection of relevant theoretical research data on GAN and its variants, then implement IDS based on these models with existing GAN models, run and evaluate, and finally collect and analyze derived measures based on confusion matrix.

1.3 Research Questions

This paper focuses on the following issues:

- What are the factors that affect performance of GAN and BiGAN based IDS?
- What are the effects of these factors?
- How can these factors be applied to improve IDS performance?

1.4 Organisation of the Thesis

The following is an overview of the structure of this thesis. The second chapter introduces the background information of relevant research. The third chapter introduces the research methods of this paper. The fourth chapter introduces the research results and analysis. And the fifth chapter introduces the discussion and conclusion of this paper.

Chapter 2

Background

In this chapter, relevant researches related to the following topics will be elaborated, including intrusion detection using deep learning techniques, GAN and its variant BiGAN, and anomaly detection based on GAN. IDS is an important tool for network system to detect security holes in the network. 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). Misuse detection has high detection accuracy and low false alarm rate when dealing with known attacks, but its performance drops sharply when facing unknown attacks and new attacks. In contrast, anomaly detection behaves excellent at handling unknown and new attacks. According to Symantec corporation's the annual Internet Security Threat Report (ISTR), hundreds of millions of new malware variants have been discovered every year in recent years, with 670 million found at the peak in 2017 alone. In addition, 1 in 10 URLs was defined malicious in 2018, compared with 1 in 16 in 2017. Because of the large number of new malicious attacks that appear all the time, a lot of scholars are now more focusing on anomaly detection [16].

The development of ADIDS usually consist of two phases, first using statistical or knowledge methods to learn and build behavior model that is considered normal from normal traffic profiles in the training phase, and then using new datasets to establish the system's reliable generalization ability against unknown intrusions in the testing phase. When the ADIDS performing a detection task, any behavior that

is observed to deviate significantly from the normal behavior model is regarded as anomaly, which can also be called an intrusion. This technology presupposes that malicious behavior is different from typical user behavior.

Nowadays, a variety of machine learning algorithms have been used to learn from intrusion datasets to create ADIDS. Machine learning is the process of extracting patterns from large amounts of data. Machine learning models consist of a complex set of "transfer functions" that can be used to identify or predict behavior [17]. Using machine learning algorithms can effectively improve the accuracy of detection and reduce the requirement of human knowledge. Machine learning algorithms generally can be classified into two categories: supervised learning and unsupervised learning.

Supervised learning algorithms learn and establish data patterns by identifying relevant features and categories of marked training data. In supervised learning IDS, each record is a pair consisting of a data source and a label that defines the record as either intrusive or normal. 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 [18]. The supervised machine learning algorithms that have been applied to intrusion detection including: Decision Tree (J. Wang et al., 2009; Rutkowski et al., 2014; Jabbar et al., 2016 [19–21]), Naïve Bayes (Yang et al., 2012; Koc et al., 2012 [22, 23]), Genetic Algorithms (Hoque et al., 2012; Murray et al., 2014 [24, 25]), Artificial Neural Network (Wang et al., 2010 [26]), Fuzzy Logic (Elhag et al., 2015 [27]), Support Vector Machines (Li et al., 2012; Y. Chang et al., 2017 [28, 29]), K-Nearest Neighbors (Lin et al., 2015 [30]) and so on.

Unsupervised learning algorithms create joint density models from a set of random variables without class labels and obtain 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. Common unsupervised learning algorithms include K-means (Annachhatre et al., 2015 [31]), Self-Organized Maps (K. Labib et al., 2002; H. G. Kayacik et al., 2007 [32,33]) and Hierarchical Clustering (Alcaraz, 2018; Shen et al., 2018 [34,35]).

Researchers working on IDS based on machine learning algorithms have also developed many methods to improve system performance, such as semi-supervised learning between supervised and unsupervised learning. It can be combined with the performance of a small number of tagged data classifiers to effectively reduce the time and cost required when applied to IDS. At present, many previous works have proposed many different semi-supervised learning techniques, such as self-training (Blount et al., 2011; Lyngdoh et al., 2018 [36,37]), co-training (Rath et al., 2017 [38]), semi-supervised SVM (Ashfaq et al., 2017 [39]), graphbased methods (Sadreazami et al., 2018 [40]), etc. In addition, multiple machine learning algorithms are combined to achieve better predictive performance using integrated approaches such as enhanced integration, bagging integration, and stack integration. Random Forest (J. Zhang et al., 2008; Jabbar et al., 2017 [41,42]) is the best example of such a method applied.

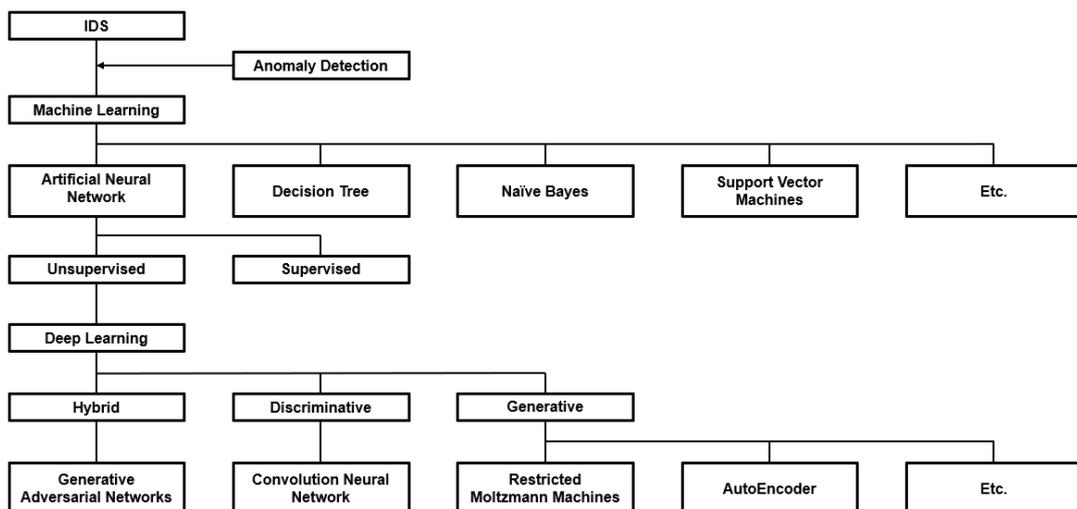


Figure 2.1: Machine Learning, Deep Learning and GAN

2.1 Deep Learning in intrusion detection

In the past few years, deep learning algorithms have developed rapidly and a lot of attention has been paid to them. As shown in Figure 2.1, deep learning is a subset of Artificial Neural Network (ANN) algorithms in the classification of machine learning algorithms. Since 1943, when McCulloch et al. [43] first proposed the idea of mathematical model based on imitating biological neurons, ANN has gone through a long process of perfection. Hebb (1949) [44] proposed the adjustment weight to lay a foundation for the subsequent improvement, then Rosenblatt et al. [45] constructed the first learnable ANN - the Perceptron in 1958, and finally in 1986 Rumelhar and Hinton et al. [46] proposed the back propagation algorithm (BP) to perfect and shape ANN. Now ANN has become one of the most widely used machine learning algorithms. ANN consists of several layers of neurons, each of which is usually connected to all neurons in the adjacent network layer through adaptive weights, which is also called full connection. According to the universal approximation theorem in ANN's mathematical theory, when the gradient of activation function does not vanish, ANN with a single hidden layer can approximate every continuous function that maps real interval to some real output interval. In other words, the single hidden layer ANN can characterize any nonlinear continuous function. This kind of single hidden ANN can be classified as shallow learning. In shallow learning, feature selection is processed separately instead of as part of ANN, which is also the common deficiency of all the machine learning algorithms mentioned above.

Deep learning was first realized by Hinton, one of the creators of BP algorithm, in 2006 [47], which also started the trend of rapid development of deep learning. Deep learning is a set of machine learning algorithms that represent data as abstract concepts corresponding to nested nonlinear hierarchy in neural networks by extending the deep neural network architecture into multiple hidden layers and attempting to learn at those levels. These levels correspond to concepts at different levels, where lower-level concepts can define higher-level concepts, and the same lower-level concepts can be used to define many higher-level concepts [48]. The biggest difference between deep learning and shallow learning is that deep learning architecture 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. Although running the deep learning algorithm requires huge computing power, due to the progress of the graphics processing unit (gpu) and deep learning can optimize the million-level parameter model into small manageable chunks through independent training of different layers in the model [47], which greatly reduces the required computing resources.

Deep learning can be classified into three categories: deep networks of unsupervised or generative learning, deep networks of supervised or discriminative learning, and hybrid deep networks. The anomaly detection technology based on generative deep learning uses the inherent attributes of unlabeled data instances to detect outliers, and automatically labels these data samples to generate labeled data. The most common generative deep learning algorithms are Restricted Boltzmann Machines (RBMs) (Hinton, 2007; Fischer et al., 2014 [49,50]) and AutoEncoder (AE) (Hawkins et al., 2002; Ranzato et al., 2007; Vincent et al., 2010 [51–53]). They all have the function of reconstructing the input data, the difference is that the RBMs is an undirected model, the data can flow in both directions in a model, which means that the RBMs will reconstruct the input by feeding it backward through the model after finishing training the data and feeding it forward, and the AE divides its hidden layer into the encoder and decoder two parts, which the decoder is responsible for the reconstruction, and the two parts will training together to minimize the discrepancy between the input data and the reconstructed data. In addition, both RBMs and AE can be stacked in layers to create deep learning models with deeper neural networks. For example, Deep Belief Network (DBN) apply layer-by-layer greedy learning strategy to stack RBMs so that many hidden layers can effectively train data by activating an RBM. And stacked autoencoders uses multi-layer AE to compress information gradually after series training.

The supervised deep anomaly detection technology learns the separate boundary from the posterior distributions of classes conditioned on the labeled training data and establishes multi-class classifier model, which is then used to classify the testing

data instances into normal class and anomaly class. Compared with unsupervised learning, supervised learning method has better performance, but lower popularity due to the lack of clean data labels. The typical representative of this kind of algorithm is the Convolutional Neural Network (CNN) (LeCun et al., 1990; 1998 [54, 55]), one of the most popular deep learning algorithms at present. CNN can extract complex hidden features from high-dimensional data with complex structure through three different layers (convolution layer, pooling layer and classification layer) that constitute it, so that it can acquire the ability to detect outliers of sequence data and image data. This makes CNN highly praised in the field of analyzing visual images. CNN-based IDS needs to convert the input data into two-dimensional pixel arrays. These arrays often shown as grayscale images, which is one of the biggest challenges for IDS of this type.

It is worth mentioning that another popular deep learning algorithm, Recurrent Neural Networks (RNN) (Pollack, 1990; Williams, 1989 [56, 57]), can be used for either unsupervised or supervised. RNN is capable of handling sequential input data of variable length. It processes one unit of input at a time and uses the output of the hidden layer as an additional input to the next input, thereby modeling the input or output of independent elements composed of sequences. This also enables RNN to capture the feature of time or space sequence data and avoid the risk of losing network state. Therefore, RNN has incomparable advantages in predicting sequence data. In addition, the introduction of the long short-term memory (LSTM) (Hochreiter et al., 1997 [58]) unit and the gated recurrent unit (GRU) (Cho et al., 2014 [59]) makes up for the limitation that RNN loses its ability to capture as the time step increases.

Hybrid deep learning combines generative models and discriminative models. In the early stage, the feature selector in the hidden layer of the hybrid model, with the help of the generative learning method, successfully obtained rich representative features from the pre-training on large dataset, which makes the hybrid model effectively reduced the sparsity issue when processing the high-dimensional data (also known as the curse phenomenon), and eliminated the irrelevant features that would cover up the anomalies. Then, in the later stage, it uses the discriminative method to

distinguish the data. There are numerous hybrid deep learning algorithms, among which the Generative Adversarial Networks used in this thesis is the most promising one.

2.2 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. GAN use the concept of adversarial learning instead of maximum likelihood, which solves the deficiencies of other generative models.

GAN consists of two networks: the generator G and the discriminator D . The generator G learns to generate real-like fake samples by transforming noise variables z into sample $G(z)$ to deceive the discriminator, whereas the discriminator D is trained to maximize the probability of determining whether its inputs are training examples or $G(z)$. Both G and D self-improve during the process so call 'adversarial', and the adversarial learning situation can be given by the following expression:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[1 - \log D(G(z))] \quad (1)$$

where $V(D, G)$ denotes binary cross entropy function for binary classification problems, $P_{data}(x)$ is the real data distribution and $P_z(z)$ is the noise variable.

Since the final aim of the GAN model is to classify real samples or fake samples by D , $V(D, G)$ is the best choice of the objective function in the classification problem. G maps noise z from latent space to input data that will be presented to D , which accepts the input data and distinguishes whether the sample comes from real data or from G . The $[1 - \log D(G(z))]$ term indicates that if D determines that the sample comes from real data, D will maximize its output, while if D determines the sample

from G, D will minimize its output. At the same time, G tries to maximize D's output when submitting the fake samples generated by G to D, so as to achieve the effect of confusing D's discrimination. As a result, D and G respectively attempt to maximize and minimize $V(D, G)$, thus forming the minimax adversarial relation in the GAN expression equation. The GAN illustration is shown in Figure 2.2.

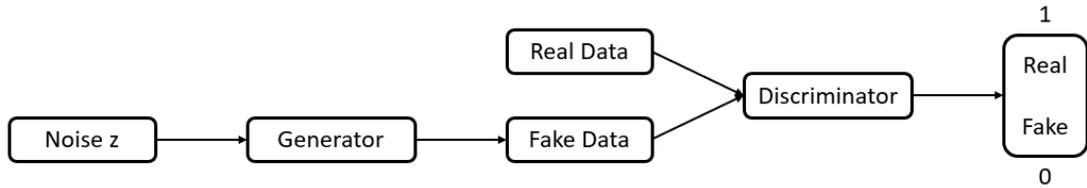


Figure 2.2: Generative Adversarial Networks

According to the theoretical derivation of Goodfellow et al [6], 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 optimization is achieved, an equilibrium state will occur between G and D, and D's output is 0.5. 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 discrepancy between the generated data distribution and the real data distribution in an implicit way through D, and then learn to reduce the discrepancy.

Although GAN has certain advantages and theoretical support, but due to practical problems and theoretical assumptions cannot be realized, many deficiencies have been found, such as the capacity issue of discriminator and the lack of the ability of learning the inverse mapping. Therefore, there are many researches developed various GAN variants to solve these problems by changing the objective function, the structure, etc. BiGAN is one of the variants with structure deformation.

2.3 Bidirectional Generative Adversarial Networks

BiGAN is a novel unsupervised feature learning framework proposed by Donahue et al. [60] in 2016, which extended by introducing an inference network into the standard GAN model so that the discriminators can not only consider inputs from data space but also jointly from data and latent space. It combines autoencoder structure into regular GAN framework to learn the latent representation. BiGAN consists of three neural networks: the generator G , the encoder E and the discriminator D , where G and E constitute the autoencoder. The encoder E compresses real data samples x into latent representation z , while G , which can also be regarded as a decoder, reconstructs the encoded data into the original data x . This autoencoder structure can reconstruct the original data by learning the joint posterior distribution $p(x|z)$, which improves the stability of the model by reducing the mode collapse caused by GAN being unable to inference to the mapping of real data to latent data. And by using the encoder to learn the inference $X \rightarrow Z$, where X represents the real data spaces and Z represents the latent spaces, of the latent representation of high-dimensional data spaces, the autoencoder can also help to handle operations at the level of abstraction. As for the benefits of learning the latent representations, complex modifications can be made more easily in the data space by means of interpolation or cascading conditions. By simultaneously training G and E as a whole part as a generative model of the adversarial learning model, the autoencoder can learn the inference while still generate high-quality samples. While D , as the other part of the adversarial learning model, receives samples from X and Z , and it has to discriminate not only the real and fake samples, but also the joint pairs $(G(z); z)$ and $(x; E(x))$. According to previous works of Zenati et al. [60–62], the BiGAN training objective function can be defined as:

$$\begin{aligned}
 & \min_{G,E} \max_D V(D, E, G) \\
 &= E_{x \sim p_X} \left[E_{z \sim p_{E(\cdot|x)}} [\log D(x, z)] \right] + E_{z \sim p_Z} \left[E_{x \sim p_{G(\cdot|z)}} [1 - \log D(x, z)] \right] \\
 &= E_{x \sim p_X} [\log D(x, E(x))] + E_{z \sim p_Z} [1 - \log D(G(z), z)] \quad (2)
 \end{aligned}$$

where $p_X(x)$ and $p_Z(z)$ are the distribution over the data and the latent representation, and $p_{EX}(x, z)$ and $p_{GZ}(x, z)$ are the joint distributions of $p_E(z | x)$, $p_X(x)$ and

$p_G(x | z)$, $p_Z(z)$ that modeled by the generator and encoder.

Unlike the generator in GAN, E and G in BiGAN learn the joint probability distribution of real data x and latent data z , instead of learning the real data distribution directly. E maps x to the latent feature space of the model, and G maps noise z to the real data space. E and G work together to try to maximize D's output close to 1, while D still tries to maximize the objective function $\min_{G,E} \max_D V(D, E, G)$ as it does in GAN. This is the minimax relation shown in Equation 2 and Figure 2.3.

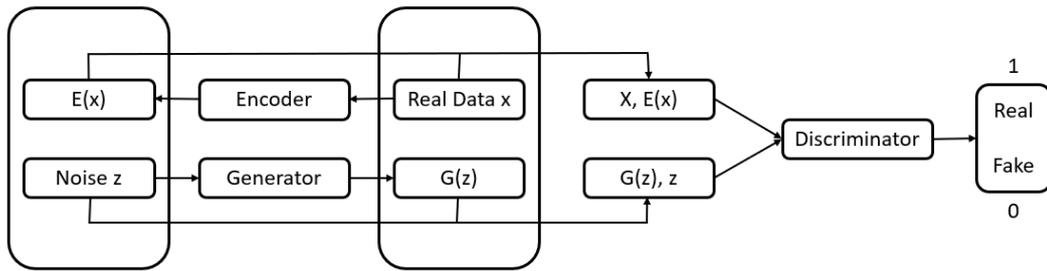


Figure 2.3: Bidirectional Generative Adversarial Networks

The optimal structure of BiGAN is similar to the optimal structure of GAN framework, but also different. The largest difference is that it optimizes the Jensen-Shannon Divergence (JSD) between the joint distribution of data space and latent feature space. Donahue et al. deduced several theorems about this. First, if and only if the joint distribution of $p_{EX}(x, z)$ and $p_{GZ}(x, z)$ are the same, and the JSD value between them achieve the global minimum, that is, E and G reach the optimum, and the output value of D is 0.5. Second, when the objective function of E and G given the optimal discriminator, the JSD can be rewritten as a ℓ_0 loss function. Since the ℓ_0 loss function does not make any assumptions about the structure or distribution of the data itself, it can infer that all structural properties of BiGAN are learned as part of the discriminator. Finally, theoretically, when E and G are optimum, E and G are inverting each other in any case, but in practice, E and G may never reach the optimum state due to the non-convexity of optimization [60].

Chapter 3

Proposed Methods

Despite GAN developed rapidly and was widely used in various aspects. However, the development of GAN-based IDS in the field of network security has just started. Currently, there are few researches on the development of IDS based on GAN and its variants algorithms, such as Efficient GAN Based Anomaly Detection (EGBAD) developed by Zenati et al (2018) based on BiGAN, and IDSGAN developed by Lin et al (2019) based on Wasserstein GAN [62–64]. Therefore, this thesis attempts to explore the factors that can affect the performance of GAN-based IDS and understand what impact can these factors have, so as to help future developers to develop more efficient GAN-based IDS more easily.

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 BiGAN are implemented by using the programming language Python with Keras as the deep learning framework. 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 eriklindernoren on GitHub [65]. The whole experiments ran on Ubuntu 18.04 system with a 2G Nvidia GeForce GTX 1050 GPU.

3.1 Data Pre-processing

The NSL-KDD dataset is a refined version of the KDD'99 dataset [66]. The KDD cup dataset has been widely used as a benchmark dataset in NIDS evaluation for many years, but it has a fatal 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. These redundant records makes the learning algorithm biased against frequent attack records but neglects infrequent but harmful attack records. And the large number of frequent normal records also weaken the learning 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.

The NSL-KDD dataset contains 41 features and one 'label' feature, and can be classified into five classes. These classes consist of normal data and four types of attacks, including Dos, Probe, R2L, and U2R. In the 41 features, three of them are categorical except the , six are binary and 32 are numeric. For categorical features, the LabelEncoder is used to transform categories to numbers and then used One-Hot-Encoding to transform all these categorical features into binary features. After encoding, we obtained a total of 123 features for training set and 117 features for testing set. Therefore, we added the 6 lacked categories as empty columns to testing set.

The datasets used in this work is from two files in the NSL-KDD dataset: KDDTrain+ and KDDTest+, which respectively contain the full NSL-KDD training set and testing set, and the dataset files are from the UNB website [67]. The training set contains 22 attack classes and one normal class. The testing set contained 37 attack classes, 21 of them from 16 training set and 16 are novel ones, and one normal class. Specific datasets details are shown in Table 3.1.

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

Table 3.1: Records Distribution of Training Set and Testing Set in NSL-KDD Dataset

3.2 Experiments Models

According to the GAN and BiGAN models provided by eriklindernoren based on Keras framework, corresponding IDS are implemented. IDS system can be divided into two stages. The first stage is to train GAN and BiGAN models. 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 normal traffic. In the IDS based on GAN, these generated examples will be entered into D along with normal traffic examples, that have been de-labeled and numeric converted, from KDDTrain+ dataset. The generated examples in the IDS based on BiGAN will be combined with the noise variables to joint pairs and then send into D with another set of joint pairs consisting of real normal traffic examples and encoded data generated by the encoder compressing real normal traffic examples. Then, D will discriminate these traffic records to complete a training process. After this training process is repeated for a given number of times, the system moves to the second stage. In this stage, all data in the KDDTest+ dataset, including normal traffic examples and malicious traffic examples, are pre-processed as training data did before, be de-labeled and digitized. These pre-processed data x_{text} will be sent to D for anomaly detection in GAN based IDS, which are different in the other IDS. In the BiGAN based IDS, x_{text} will be compressed by the encoder to encoded data $x_{encoded}$, and then combined with $x_{encoded}$ to joint pair $(x_{encoded}, x_{text})$, which will be sent to D for detection instead of x_{text} . Algorithm 1 and Algorithm 2 shows the outlines of the IDSes based on GAN and BiGAN.

Algorithm 1: IDS based on GAN

Input: Original normal traffic examples x_{normal} from the training set
KDDTrain+; The random variable noise n ;

Output: Detection result

Initialize the generator G and the discriminator D ;

for *Initialized GAN* **do**

- for** $i = 1, \dots, \text{training times}$ **do**
 - for** G **do**
 - G generates the fake normal traffic examples $x_{generated}$ from n
based on x_{normal} ;
 - Send $x_{generated}$ to D ;
 - 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;

end

Algorithm 2: IDS based on BiGAN

Input: Original normal traffic examples x_{normal} from the training set
KDDTrain+; The random variable noise n ;

Output: Detection result

Initialize the generator G , the encoder E and the discriminator D ;

for *Initialized BiGAN* **do**

- for** $i = 1, \dots, \text{training times}$ **do**
 - for** G **do**
 - G generates the adversarial normal traffic examples $x_{generated}$ from n based on x_{normal} ;
 - Send joint pair $(n, x_{generated})$ to D ;
 - end**
 - for** E **do**
 - E compresses x_{normal} into latent space by encoding x_{normal} to $x_{encoded}$;
 - Send joint pair $(x_{encoded}, x_{normal})$ to D ;
 - end**
 - for** D **do**
 - D classifies dataset including joint pairs $(n, x_{generated})$ and $(x_{encoded}, x_{normal})$;
 - end**
- end**

end

for *Trained D* **do**

- E encodes all traffic examples x_{test} in the testing set KDDTest+ to $x_{test(encoded)}$;
- D classifies the joint pair dataset of $(x_{test(encoded)}, x_{test})+$, getting predicted labels;

end

In the detection stage, the trained discriminator D can distinguish normal data from abnormal data and output the output value of the range from 0 to 1. When the output value is 1, D determines that the input is normal traffic example through the prior adversarial learning in the training stage, otherwise the input is malicious.

Predicted labels are generated by D after the testing dataset is input into D. All values that are not equal to 1 in the predicted labels will be classified as 0, meaning that the label of the data is attack.

3.3 Evaluation Methods

In order to comprehensively evaluate the performance of IDS based on GAN and BiGAN, various parameters of the two IDSes were adjusted for running and testing the systems. The whole evaluation experiments were phased to evaluate the performance of IDSes with different value of learning rates, sample sizes (batch size), training times (epochs), and the number of neural network layers (including the encoder, the generator, and the discriminator). Among them, the influence of learning rate was tested in one phase, batch sizes and epochs were tested in one phase, and numbers of layers of neural networks were tested in one phase. The parameters tested in the same phase are related to each other.

In the first phase of the experiment, the influence of hyper-parametric learning rate of the IDSes' performances will be observed in the way that the initial value of the learning rate is 0.0001, and the value decreases by 0.000001 each time when the system completes a intrusion detection mission, and finally reduces to 0.000001. In the next phase, the impact of an order of magnitude change in batch sizes and epochs on the IDS models will be evaluated. The order of magnitude changes by the power of 10 at each time of testing, so that the IDS performance variation from small values to large values of these parameters can be shown more intuitive. The final phase evaluates the efficiency of the IDS models by running in combinations of generators, encoders, and discriminators with different numbers of hidden layers. As the tested neural networks start from the shallow neural network with only one hidden layer, and gradually increase the hidden layer until the relatively deep neural network with 20 hidden layers, the effect of the depth of neural network on the detection efficiency is observed.

3.4 Evaluation Metrics

IDS has many classification metrics. Since the experiments in this work is mainly to discriminate normal records and malicious attack records, a confusion matrix of two-classes classifiers is adopted to calculate the performance metrics. 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.

Based on the above situations, the confusion matrix of an IDS is shown in Table 3.2:

		Predicted	
		Normal	Anomaly
Actual	Normal	TP	FN
	Anomaly	FP	TN

Table 3.2: Confusion Matrix for the IDS

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

- Accuracy: The proportion of all predicted instances, including normal or anomaly, that are correctly be predicted by the IDS. It is one of the longest

used metrics to measure IDS performance, but it is not very useful when the classes are not balanced.

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

- Precision: The ratio of normal records that are correctly discriminated by the IDS to all records that IDS identified those as normal.

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

- Recall: The percentage of all normal records correctly identified by IDS.

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

- F1-score: The balance between Precision and Recall, and it is expressed as the harmonic mean of the two metrics.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- AUC: The value of AUC is the sum of the area under the RoC curve, which plots the False Positive Rate (FPR) on the x-axis and the True Positive Rate (TPR) on the y-axis. The FPR and TPR are respectively the ratio of the number of normal records that are identified as attack records and the number of attack records that correctly be predicted to the total number of actual attack records. They can be expressed as follows:

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

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

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, F-1 score becomes an important indicator, the higher the score, the better performance that the IDS has.

Chapter 4

Results and Analysis

All experiments are done on a physical machine. For the accuracy of the data, each IDS model corresponding to different parameters is run 50 times, and the final result is the average of the 50 runs.

This chapter will present all experimental results and corresponding analysis results.

4.1 Performance of GAN Based IDS in Different Value of Learning Rate

First of all, we evaluate the impact of learning rate. Learning rate is a non-negligible configurable hyper-parameter in neural network training, with its value ranging from 0.0 to 1.0. It determines how quickly the model adapts to the learning, and too large or too small of its value can cause serious issues. If the value of learning rate is too large, the model will converge to the sub-optimal solution too quickly and miss the optimal solution, while if the learning rate is too small, the process will take too long and get into stuck. Therefore, it is very important to choose an appropriate learning rate.

As shown in the Figure 4.1, when the learning rate starts to decrease from 0.0001, the AUC value of IDS maintains a relatively stable range from 0.7 to 0.8 until the

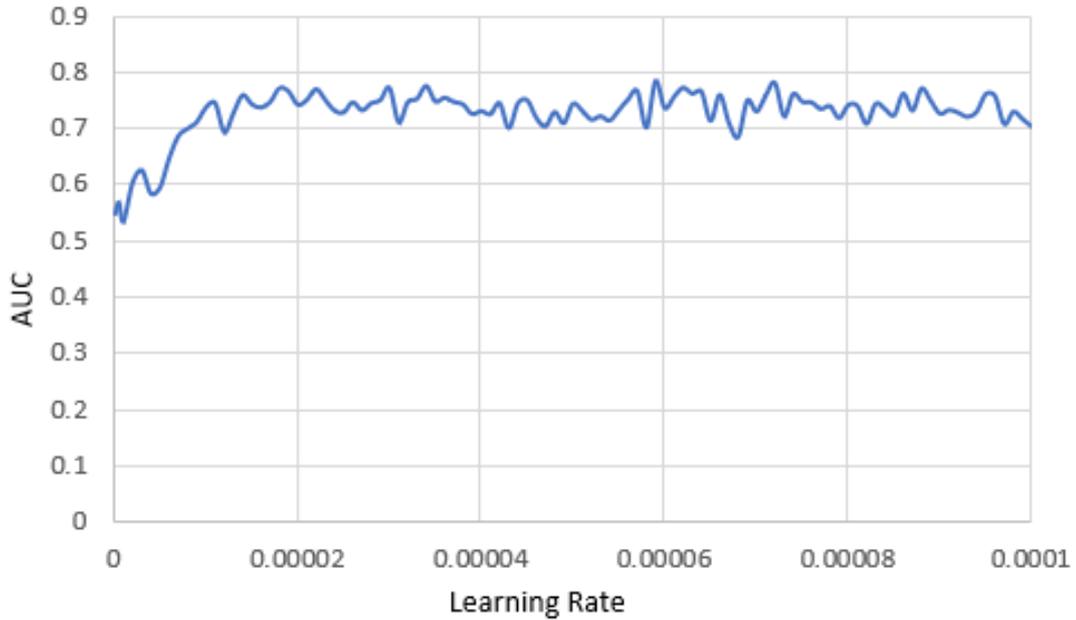


Figure 4.1: Value of AUC through learning rate

learning rate reaches about 0.00001, and then sharply drop to nearly 0.5. Therefore, in the following experiment, the value of learning rate is taken as a relatively compromised value of 0.00005.

4.2 Performance of IDSes with Different Amounts of Batch Sizes and Epochs

It is not enough to transmit the complete dataset in the neural network once, that is, only one epoch is not enough. When the number of times of transfer is not enough, the gradient curve of the model is underfitted, so the complete dataset needs to be transferred multiple times in the same neural network to reach the optimal solution. However, if the value of epochs is too large, over-fitting will be occurred. Batch size refers to the size of several new small datasets that need to be divided into when a dataset is too large to be processed through the neural network at one time. Continuously increasing the batch size by an order of magnitude makes it possible to test IDS's capability of handle big data. In addition, the sample data

with batch size was extracted randomly from the KDDtrain+ dataset in this project, which increased the complexity of the dataset and thus improved the precision of evaluations of the IDSes.

Sample Size	Training Times	Accuracy	Precision	Recall	F1	AUC
10	10	0.63904143	0.451381098	0.415044795	0.406822691	0.611794586
	100	0.626120032	0.536543401	0.654582432	0.562117701	0.629582185
	1000	0.513542406	0.472723305	0.973622696	0.634984526	0.569506353
	10000	0.504934794	0.467827171	0.990485017	0.634548298	0.563996892
100	10	0.616064141	0.463286369	0.386618268	0.389568863	0.588154455
	100	0.69586808	0.63315869	0.85690969	0.714014688	0.715457105
	1000	0.479009936	0.452086354	0.982370508	0.619123491	0.540238476
	10000	0.46674725	0.446677238	0.994629801	0.616464766	0.530958632
1000	10	0.626288591	0.463101253	0.428905365	0.394287943	0.60227899
	100	0.628694996	0.565064479	0.923190197	0.690332408	0.664517252
	1000	0.513364975	0.47406828	0.977988879	0.636735257	0.569881605
	10000	0.44722099	0.437901458	0.998187622	0.608737742	0.514240308
10000	10	0.655267477	0.616196151	0.453552672	0.478903897	0.630730984
	100	0.632880145	0.56684993	0.913659767	0.690749568	0.667034045
	1000	0.517825142	0.475727539	0.981278962	0.639284016	0.574199444
	10000	0.438480305	0.434072156	0.999330656	0.605247224	0.506701874

Table 4.1: The performance of GAN based IDS with different sample sizes and training times

Sample Size	Training Times	Accuracy	Precision	Recall	F1	AUC
10	10	0.647385114	0.468160934	0.321640408	0.35634656	0.607761683
	100	0.710794446	0.607160003	0.719689012	0.648038571	0.711876377
	1000	0.59780873	0.527459879	0.967377201	0.679571622	0.642762862
	10000	0.472167761	0.449987805	0.99395016	0.619182022	0.535637123
100	10	0.642614443	0.50829326	0.308191741	0.345576757	0.601935425
	100	0.71725071	0.644590582	0.856595613	0.728954693	0.734200557
	1000	0.545896913	0.494190792	0.980836165	0.654594015	0.598802716
	10000	0.488134315	0.458205437	0.997415302	0.627525612	0.550083012
1000	10	0.624784865	0.471567214	0.359875399	0.377514115	0.592561404
	100	0.66527901	0.585482102	0.90506642	0.705614673	0.694446636
	1000	0.542982612	0.493987768	0.983117084	0.654477582	0.596520359
	10000	0.500563343	0.464221949	0.996282566	0.632941071	0.560862393
10000	10	0.635710167	0.500316578	0.439903203	0.416048138	0.611892301
	100	0.670774929	0.589336428	0.923066625	0.713786947	0.701463571
	1000	0.471934883	0.449486234	0.992050252	0.618468759	0.535201468
	10000	0.481660309	0.453777678	0.99617959	0.623503748	0.544246189

Table 4.2: The performance of BiGAN based IDS with different sample sizes and training times

According to Table 4.1 and Table 4.2, the size of training dataset does not show a uniform trend on the performance of IDSes based on GAN and BiGAN, but the time spent by running the models increases as the size training dataset grows, and there is no fixed rule under cross-reference. AUC values of the IDSes' evaluations, for example, although with the growth of the value of the batch size, most results present the tendency of increasing, but increasing processes are complicated, some

of them rise after a drop, and some rise before they drop and then rise again, and finally the growth of the AUC value was slightly, and some metrics show a continuous decline trend. And considering the cost of time: when the value of the batch size was 10, the IDS models complete 50 times evaluations within hours, and when this value increased to 10000, the models took days to measurement, obviously this is a very low efficiency.

But for epochs, its value has a significant impact on IDSes performance. Obviously, there is a hundred place threshold for the effect of training times on model efficiency. For both GAN-based and BiGAN-based IDS, when epochs is 100, the value of AUC, accuracy, and precision in their evaluation reports are all reached maximum and then slowly decrease, which means that the model performs best at this point. Most of the time, the F1 score shows a similar trend, except when the batch size is 10. Moreover, the value of recall increased with the growth of epochs and finally infinitely close 1. For more details, a micro-detailed test of the impact of the change in the parameter was conducted to observe the change in the AUC of the evaluations of IDSes from the value of epochs is 10 to 1000. The trends of the test are shown in below.

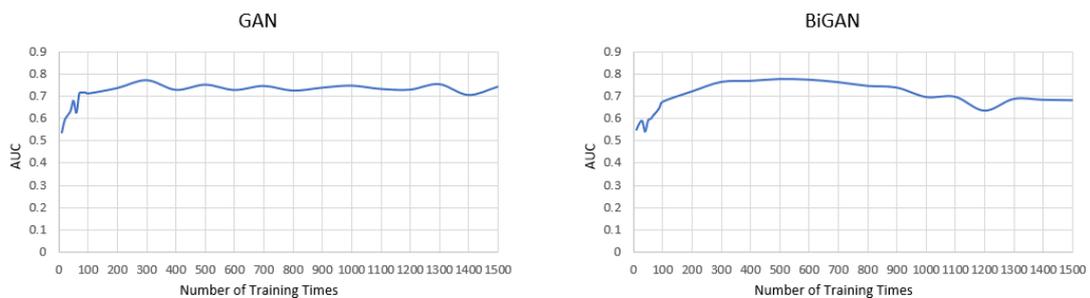


Figure 4.2: Value of AUC through increased value of epochs

As shown in the Figure 4.2, the AUC values in the evaluations of IDSes based on GAN and BiGAN reached the peak value when the epoch was 300 and 500, respectively, and then both of them gradually declined. In contrast, the value of AUC in the performance of GAN-based IDS drops more slowly but is more volatile, while the value of AUC in the BiGAN one has a wider peak range, which means that the BiGAN-based IDS performs better than the GAN-based IDS in the peak range and also works more stable.

4.3 Performance of IDSes with Different Number of Layers of Neural Network Hidden Layers

The meaning of hidden layer in neural network is to abstract the features of input data into another dimension space to show its more abstract features, which can be better divided linearly. Multiple hidden layers are actually multi-level abstractions of input features, and the ultimate purpose is to better linearly partition different types of data. Theoretically, the more number of hidden layers there are, the clearer the feature division will be. However, after a certain number of hidden layers are added, the enhancement of classification effect will be less and less obvious, and the neural network may become too complicated and even over-fitting will occur.

Before comparing the effects of generator and discriminator depth, evaluate the performance of BiGAN-based IDS with combinations of encoders and generators with different hidden layers. The encoders and generators in BiGAN make up the autoencoders, and in some variants of the more advanced version of BiGAN, the adversarial learning take place between them instead of the discriminator. Theoretically, the optimal structure of the autoencoder is when the two neural networks are inverse to each other, which means they are relatively symmetric.

Generator	Encoder	Accuracy	Precision	Recall	F1	AUC
1	1	0.590900018	0.532784076	0.880511791	0.65369417	0.626128256
	5	0.648853353	0.577804665	0.882669138	0.687173836	0.677294594
	10	0.69209324	0.603074164	0.765338276	0.659521138	0.701002731
	20	0.656717974	0.591418979	0.821001957	0.670706823	0.676701399
5	1	0.631791164	0.561339293	0.913129441	0.687959866	0.666013018
	5	0.698771292	0.625641673	0.846400989	0.696106501	0.716728898
	10	0.619490774	0.555482241	0.847368963	0.657811672	0.647209768
	20	0.671888307	0.600436363	0.823874987	0.682608143	0.690375895
10	1	0.660124645	0.583688804	0.91532283	0.705480733	0.691166831
	5	0.648314407	0.580936594	0.804654515	0.652646085	0.667331544
	10	0.689555979	0.624923369	0.824055195	0.698834066	0.7059164
	20	0.666523243	0.596404627	0.846195037	0.688262585	0.688378435
20	1	0.648782381	0.572587039	0.883240655	0.689577423	0.677301774
	5	0.680879613	0.603275928	0.853815261	0.69329519	0.701915423
	10	0.652528389	0.575570062	0.818211307	0.657764352	0.67268198
	20	0.657097232	0.615468289	0.82828236	0.677553395	0.677920109

Table 4.3: The performance of BiGAN based IDS with different numbers of layers of Generator and Encoder. The Discriminator used in this table has only one layer.

Table 4.3 shows the effect of depth changes in encoders and decoders on the per-

formance of BiGAN-based IDS. In terms of results, there is no directional trend except that as the depth of encoder increases, the value of F1 increases in a wave sharp, that is, it first grows then drop, and then raise again. It is worth noting, however, that in some cases the IDS performs well when encoder and decoder have the same number of hidden layers, especially when they both have five layers of hidden layers. While another interesting phenomenon is that if the AUC value form extracted separately, and the number of hidden layer upon layer of the encoder and decoder for axis, will find that in most cases the table is symmetry, for example: when the numbers of hidden layers in the encoder are 5 and 10, as the growth of the decoder on the number of hidden layers, the AUC value showed a trend of twists and turns of sexual growth and reduce respectively, one goes up then down and then up, and the other did the opposite way. In a word, the encoder and decoder with symmetric structure have some merits. Therefore, in the following experiments, it will be regarded as a generator of symmetric autoencoder structure in BiGAN and compared with the discriminator.

Generator	Discriminator	Accuracy	Precision	Recall	F1	AUC
1	1	0.642916075	0.584664045	0.893255072	0.694321005	0.673367191
	5	0.724363467	0.678468102	0.703171661	0.685465934	0.721785706
	10	0.607055092	0.595217259	0.255076717	0.354418261	0.564240611
	20	0.55157248	0.180338964	0.011842241	0.021116197	0.485919952
5	1	0.589879791	0.528030959	0.911569354	0.662101978	0.629009956
	5	0.727510646	0.686001565	0.682087324	0.679635989	0.721985375
	10	0.606010468	0.591426371	0.253779219	0.351966217	0.563165227
	20	0.557773687	0.214656924	0.00407785	0.007752489	0.490422389
10	1	0.604220635	0.546768548	0.890165791	0.664806645	0.639002868
	5	0.726131121	0.682109153	0.685547317	0.680445016	0.721194527
	10	0.611652768	0.601430444	0.27291731	0.371175914	0.570449148
	20	0.556400816	0.186572602	0.005056122	0.009479357	0.489335511
20	1	0.629905962	0.564166887	0.89261662	0.682989132	0.661861961
	5	0.747201029	0.698804991	0.729610751	0.709709923	0.745061356
	10	0.600623226	0.567394091	0.247559469	0.338442013	0.557676719
	20	0.555855216	0.205249433	0.012480692	0.02095339	0.489759398

Table 4.4: The performance of GAN based IDS with different numbers of layers of Generator and Discriminator

As Table 4.4 shows, in GAN based IDS, with the increase of the number layers of the generator, the experiment results are not affected by bias effect. Meanwhile, the experiment results recorded by running GAN with the increase of the number of discriminator layers show a peculiar result. It shows that when the discriminator has five layers, the AUC, precision and accuracy reach the maximum value, and then the values of these metrics continue to decline with the increase of the number of layers

Generator & Encoder	Discriminator	Accuracy	Precision	Recall	F1	AUC
1	5	0.668583659	0.621038289	0.406647101	0.481187821	0.636721821
	10	0.552648155	0.369492133	0.04025847	0.067678784	0.490321318
	20	0.566197214	0.356311996	0.000864998	0.001673818	0.497430473
5	5	0.667975958	0.652947967	0.397616105	0.483569895	0.635089515
	10	0.556897622	0.467612042	0.036504994	0.061873668	0.493597311
	20	0.568871984	0.287512692	0.000550922	0.001088232	0.499741681
10	5	0.69065605	0.759049295	0.437457522	0.532133207	0.659857102
	10	0.552559439	0.323221977	0.031505509	0.053826685	0.489178688
	20	0.568847587	0.227718416	0.000391309	0.000779749	0.499700836
20	5	0.692776348	0.748312466	0.421599217	0.529094889	0.659790492
	10	0.553714957	0.310331934	0.033853362	0.056704171	0.490479241
	20	0.568876419	0.306562327	0.000478838	0.000948965	0.499736809

Table 4.5: The performance of BiGAN based IDS with different numbers of layers of Generator and Discriminator. The Decoder (Generator) and Encoder are used together as Generator in a symmetric structure.

of the discriminator. This is highly similar to the results of running model with different number of epochs. The main difference between the two evaluation reports lies in the recall value. In the previous phase, the recall value increased with the increase of training times, while in this phase, the recall value showed an opposite performance, which decreased with the increase of the depth of the discriminator.

The records in Table 4.3 and Table 4.5 can be combined to obtain a comprehensive evaluation of BiGAN-based IDS with different levels of generators and discriminators. Accuracy and precision metrics are in peak when the number of hidden layers of the discriminator increases to 5 and then continually decreasing, which is similar to the trend in accuracy and precision in GAN based IDS evaluation report. While the value of recall and F1 score declined sharply with the increase of the number of discriminator hidden layers. Although the recall also continued to decrease in the evaluation report of IDS based on GAN, but the magnitude was not so dramatic, while the performance of F1 score was significantly different in the two reports. The value of AUC is similar to its performance in the evaluation report of GAN based IDS only when the encoder and generator level is 1. In the other conditions, it shows a continuous decline trend. However, with reference to the three tables at this phase, it is reasonable to infer that the peak range of AUC in the evaluation report of IDS based on BiGAN has moved forward. As in the previous phase, this

will be further tested. In addition, in the evaluation reports of GAN-based IDS and BiGAN-based IDS, when the number of hidden layers of the discriminator reaches 20 and 10 respectively, the AUC value is less than but close to 0.5, which usually means two situations: algorithm is bad or the discriminator is in the chance level in the long run. In this experiment, it is suitable for the second situation which means that over-fitting occurs.

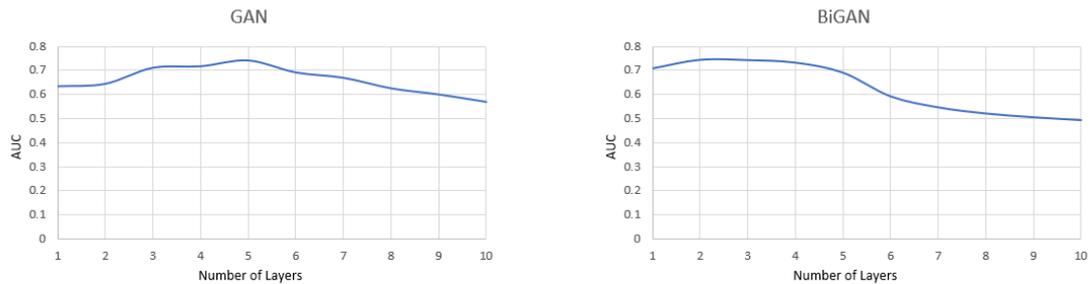


Figure 4.3: Value of AUC through increased number of layers of Discriminator’s hidden layers

As shown in Figure 4.3, the AUC value in the evaluation report of IDSes based on GAN and BiGAN reaches the maximum value as expected when the number of layers of discriminator is 5 and 3 respectively, and then gradually decreases. By contrast, the value of AUC in BiGAN-based IDS evaluation reports declines more steeply and over-fits occur earlier, but it presents a smoother curve. These mean that BiGAN-based IDS’s discriminator has less capacity but the system is more stability.

4.4 Limitations of Study

- Due to time, another GAN variant named Wasserstein GAN that mentioned in this thesis, which is applied in the field of network information security intrusion detection, has not been evaluated in this work.
- As this experiment aims to compare the influence of parameters, the detection method is relatively simple with low accuracy, which needs to be improved.

- During the experiment, the average value of the evaluation results obtained by running IDS for 10 times and the average value of the results obtained by running IDS for 50 times presented different effects. Limited by time and experimental equipment, it was impossible to further verify the results obtained by more times such as 100 times and 1,000 times.
- When the numbers of layers of encoder and generator are the same, the performance of BiGAN-based IDS is relatively outstanding, which is limited by the experimental equipment and cannot be further verified

4.5 Future Work

After evaluating and testing the parameters that affect GANs based IDS, some new considerations about how to research and develop such IDS more effectively have not yet been discussed in this paper for further study. These future direction including:

- Some parameters that may affect the performance of GANs based IDS, such as activation functions, number of neurons, and so on, can be tested in the future.
- Some other GAN variants may be used for intrusion detection. For example, the InfoGAN of the autoencoder deformation-occurring re-discriminator also uses the latent space; Cycle-GAN used in the field of images anomaly detection; And C-RNN-GAN which is combined with RNN and both these two algorithms had been apply for intrusion detection. These variants more or less show the potential in the field of network security intrusion detection.
- In addition, there are many other data sets for testing IDS besides the common NSL-KDD dataset, such as CAIDA, ISCX 2012, ADFA-LD, CICIDS 2017, etc. These data sets have their own characteristics, and testing them in GANs based IDS can be a good supplement.

Chapter 5

Conclusions

As society's reliance on information technology and related threats have increased, so has the importance of security. As an important member of information security technology, intrusion detection technology develops rapidly. Especially with the development and popularity of neural network algorithm, deep learning algorithm has been widely applied in intrusion detection. As a deep learning algorithm, generative antagonistic learning algorithm has a novel concept, broad prospects and proved itself in anomaly detection in other fields. It has great potential in the field of network security intrusion detection.

Although the potential is great, the IDS based on the algorithm of generative adversarial networks class is very few. This is undoubtedly due to the fact that this kind of algorithm is relatively new and the hardware requirements are strict, but the lack of relevant supplementary research data is also one of the most important reasons.

This work has shown the effect of parameters and neural network structure on the performance of GANs-based IDS. The experimental results have shown that epoch and the number of hidden layers of the discriminator have significant effects on IDS performance, and GAN combined with autoencoder can effectively improve IDS stability by utilizing latent space.

Bibliography

- [1] Liao H-J, Lin C-HR, Lin Y-C, Tung K-Y (2013b) Intrusion detection system: a comprehensive review. *J Netw Comput Appl* 36(1):16–24
- [2] Shone, N., Ngoc, T., Phai, V. and Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), pp.41-50.
- [3] Yin, C., Zhu, Y., Fei, J. and He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, 5, pp.21954-21961.
- [4] Lin, W., Lin, H., Wang, P., Wu, B. and Tsai, J. (2018). Using convolutional neural networks to network intrusion detection for cyber threats. 2018 IEEE International Conference on Applied System Invention (ICASI).
- [5] Lin, S. Z.; Shi, Y.; and Xue, Z.2018. Character-level intrusion detection based on convolutionalneural networks. In *Proceedings of InternationalJoint Conference of Neural Networks*, 3454–3461. Rio deJaneiro, Brazil: IEEE.
- [6] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. *NIPS*, 2014.
- [7] Luis Mart, Nayat Sanchez-Pi, Jos Manuel Molina, and Ana Cristina Bicharra Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15(2):2774–2797, 2015.

- [8] Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. arXiv preprint, arXiv:1703.05921 (2017)
- [9] Thomas Schlegl, Philipp Seebock, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. n International Conference on Information Processing in Medical Imaging, pp. pp. 146157, 2017.
- [10] Hu WW, Tan Y. Generating adversarial malware examples for black-box attacks based on GAN. arXiv preprint arXiv: 1702.05983, 2017.
- [11] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, “Efficient GAN-Based Anomaly Detection,” pp. 1–7, 2018.
- [12] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. arXiv preprint arXiv:1809.02077, 2018.
- [13] M. Salem, S. Taheri, J. S. Yuan, ”Anomaly generation using generative adversarial networks in host based intrusion detection”, CoRR, Dec. 2018.
- [14] Heaton, J. (2009). Introduction to neural networks with Java. Chesterfield (MO, USA): Heaton Research.
- [15] Y. Hong, U. Hwang, J. Yoo, S. Yoon, How generative adversarial networks and their variants work: An overview of GAN, arXiv preprint arXiv:1711.05914v7 (2017).
- [16] Symantec: 2019 Internet Security Threat Report (ISTR), vol. 24, February 2019
- [17] S. Dua and X. Du, Data mining and machine learning in cybersecurity. CRC press, 2016
- [18] Khraisat, A., Gondal, I., Vamplew, P. and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1).

- [19] Juan Wang, Qiren Yang, Dasen Ren, “An intrusion detection algorithm based on decision tree technology”, In the Proc. of IEEE Asia-Pacific Conference on Information Processing, 2009.
- [20] Rutkowski L, Jaworski M, Pietruczuk L, Duda P (2014) Decision trees for mining data streams based on the Gaussian approximation. *IEEE Trans Knowl Data Eng* 26(1):108–119
- [21] Jabbar, M. and Samreen, S. (2016). Intelligent network intrusion detection using alternating decision trees. 2016 International Conference on Circuits, Controls, Communications and Computing (I4C).
- [22] Yang, X. and Tian, Y. L. (2012). EigenJoints-based action recognition using Naïve-Bayes-nearest-neighbor. 2012 IEEE computer society, 2012, pp. 14–19
- [23] Koc, L., Mazzuchi, T. A. and Sarkani, S. (2012). A network intrusion detection system based on a hidden Naïve Bayes multiclass classifier. *Expert Syst Appl*, vol. 39, no. 18, pp. 13492–13500, 2012
- [24] Hoque MAM, Bikas MAN (2012) An implementation of intrusion detection system using genetic algorithm. *International Journal of Network Security & Its Applications* 4:2
- [25] S. N. Murray, B. P. Walsh, D. Kelliher, and D. T. J. O’Sullivan, ”Multi-variable optimization of thermal energy efficiency retrofitting of buildings using static modelling and genetic algorithms – a case study,” *Build Environ*, vol. 75, no. Supplement C, pp. 98–107, 2014
- [26] G. Wang, J. Hao, J. Ma, and L. Huang, ”A new approach to intrusion detection using artificial neural networks and fuzzy clustering,” *Expert Syst Appl*, vol. 37, no. 9, pp. 6225–6232, Sep. 2010
- [27] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, ”On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems,” *Expert Syst Appl*, vol. 42, no. 1, pp. 193–202, 2015

- [28] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Syst Appl*, vol. 39, no. 1, pp. 424–430, Jan. 2012
- [29] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," in *Proc.IEEE Int.Conf.Comput. Sci. Eng./IEEE Int. Conf. Embedded Ubiquitous Comput.*, Jul. 2017, pp. 635–638.
- [30] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl-Based Syst*, vol. 78, no. Supplement C, pp. 13–21, April 2015
- [31] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden Markov models for malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 2, pp. 59–73, May 2015
- [32] K. Labib and R. Vemuri. NSOM: A real-time network-based intrusion detection system using self-organizing maps. Technical report, Dept. of Applied Science, University of California, Davis, 2002
- [33] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20(4):439–451, 2007.
- [34] Alcaraz C (2018) Cloud-assisted dynamic resilience for cyber-physical control systems. *IEEE Wirel Commun* 25(1):76–82
- [35] Shen C, Liu C, Tan H, Wang Z, Xu D, Su X (2018) Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks. *IEEE Wirel Commun* 25(6):26–31
- [36] J. J. Blount, D. R. Tauritz, and S. A. Mulder (2011) Adaptive rule-based malware detection employing learning classifier systems: a proof of concept. *Computer software and applications conference workshops (COMPSACW)*, 2011 IEEE 35th annual, pp. 110–115

- [37] J. Lyngdoh, M. I. Hussain, S. Majaw, and H. K. Kalita (2018) An intrusion detection method using artificial immune system approach. International conference on advanced informatics for computing research, pp. 379–387: Springer
- [38] Rath, P. S., Barpanda, N. K., Singh R. P., and Panda, S.: A Prototype Multiview Approach for Reduction of False alarm Rate in Network Intrusion Detection System, IJCNCS, 5, 49–59, 2017.
- [39] Ashfaq RAR, Wang X-Z, Huang JZ, Abbas H, He Y-L (2017) Fuzziness based semisupervised learning approach for intrusion detection system. Inf Sci 378:484–497
- [40] Sadreazami H, Mohammadi A, Asif A, Plataniotis KN (2018) Distributed-graphbased statistical approach for intrusion detection in cyber-physical systems. IEEE Transactions on Signal and Information Processing over Networks 4(1):137–147
- [41] J. Zhang, M. Zulkernine, and A. Haque, “Random-forests-based network intrusion detection systems,” IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 38, no. 5, pp. 649–659, Sep. 2008.
- [42] M. A. Jabbar, R. Aluvalu, and S. S. Reddy S, ”RFAODE: A Novel Ensemble Intrusion Detection System,” Procedia Computer Science, vol. 115, pp. 226–234, 2017/01/01/ 2017
- [43] McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 1943, 5, 115–133.
- [44] Hebb, D.O. The Organization of Behavior; John Wiley Sons, Inc.: New York, NY, USA, 1949.
- [45] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. 1958, 65, 386
- [46] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. Nature 1986, 323, 533.

- [47] Hinton, G.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* 2006, 18, 1527–1554.
- [48] Deng, L.; Yu, D. Deep learning: Methods and applications. *Found. Trends Signal Process.* 2014, 7, 197–387.
- [49] Hinton, G.E.: Boltzmann machine. *Scholarpedia* 2(5), 1668 (2007)
- [50] Fischer, A., Igel, C.: Training restricted Boltzmann machines: an introduction. *Pattern Recognit.* 47, 25–39 (2014)
- [51] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.
- [52] Ranzato, M.; Huang, F.J.; Boureau, Y.L.; LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the CVPR'07 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
- [53] Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* 2010, 11, 3371–3408.
- [54] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* 1998, 86, 2278–2324
- [55] LeCun, Y.; Boser, B.E.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.E.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1990; pp. 396–404.
- [56] Pollack, J.B. Recursive distributed representations. *Artif. Intell.* 1990, 46, 77–105.
- [57] Williams, R.J. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical report, Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern . . . , 1989.

- [58] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [59] Cho, K.; Merri’enboer, B.V; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [60] J. Donahue, P. Krahenbuhl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
- [61] Thomas Schlegl, Philipp Seebock, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *n International Conference on Information Processing in Medical Imaging*, pp. pp. 146157, 2017.
- [62] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, “Efficient GAN-Based Anomaly Detection,” pp. 1–7, 2018.
- [63] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. *arXiv preprint arXiv:1809.02077*, 2018.
- [64] Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.
- [65] GitHub. (2018). eriklindernoren/Keras-GAN. [online] Available at: <https://github.com/eriklindernoren/Keras-GAN/blob/master/gan/gan.py>.
- [66] Kdd.ics.uci.edu. (2018). KDD Cup 1999 Data. [online] Available at: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- [67] Unb.ca. (2018). NSL-KDD — Datasets — Research — Canadian Institute for Cybersecurity — UNB. [online] Available at: <http://www.unb.ca/cic/datasets/nsl.html>. generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.