A WASSERSTEIN GAN BASED FRAMEWORK FOR ADVERSARIAL ATTACKS AGAINST INTRUSION DETECTION SYSTEMS

by

Fangda Cui

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2022

To my great parents, for all the love, support, and encouragement you have provided, and for inspiring me to be a better person!

Table of Contents

List of Tables . . . . **v**

List of Figures . . . . **vi**

Abstract . . . . **viii**

Acknowledgements . . . . **ix**

Chapter 1 Introduction **1**

1.1 Intrusion Detection 1

1.2 Adversarial Attacks 3

1.3 Overview of Wasserstein GAN Based Framework 4

1.4 Thesis Outline 5

Chapter 2 Related Work **6**

2.1 Intrusion Detection Systems: From Signature to Deep Learning . . . 6

2.2 Generative Adversarial Network 9
2.2.1 Wasserstein GAN 11
2.2.2 Conditional GAN 12

2.3 Adversarial Attacks against Intrusion Detection Systems 13
2.3.1 White-box Adversarial Attacks 13
2.3.2 Black-box Adversarial Attacks Using GAN 14

Chapter 3 Wasserstein GAN Based Adversarial Attacks **16**

3.1 Selected Schemes for IDS 16
3.1.1 Machine Learning 16
3.1.2 Multilayer Perceptron 17
3.1.3 Convolutional Neural Network 17
3.1.4 Recurrent Neural Network 19

3.2 Wasserstein GAN Based Framework 21

3.3 Conditional Wasserstein GAN Based Framework 24
3.3.1 Input Conditions 24
3.3.2 From WGAN to CWGAN 26

Chapter 4 Performance Analysis **29**

4.1 CICIDS2017 Dataset 29
4.1.1 Dataset Selection 29
4.1.2 Dataset Description 30
4.1.3 Data Preprocessing 31
4.1.4 Functional Features 32

4.2 Experimental Setup and Evaluation Metrics 32

4.3 Experimental Results 34
4.3.1 Performance of Selected IDS Schemes 34
4.3.2 Performance of Wasserstein GAN Based Framework 36
4.3.3 Performance of Conditional Wasserstein GAN Based Framework 41
4.3.4 Results Summary and Discussion 50

Chapter 5 Conclusion and Future Work **53**

5.1 Conclusion 53

5.2 Future Work 53
5.2.1 Other Deep-generative algorithms 53
5.2.2 Dissimilarity Measurement 56
5.2.3 Federated Learning and GAN 57
5.2.4 GAN-based Data Synthesis for IDS training 59

Bibliography **60**

List of Tables

3.1 Description of the architecture of the Multilayer Perceptron for the IDS. 18

3.2 Description of the architecture of the Convolutional Neural Network for the IDS. 20

3.3 Description of the architecture of the Recurrent Neural Network for the IDS. 20

3.4 Architecture of the generator and discriminator in the WGAN framework. 23

3.5 Architecture of the generator and the discriminator in the conditional WGAN framework 27

4.1 Type of attack traffic records in CICIDS2017. 31

4.2 Characteristic features and corresponding attack types for CICIDS2017. 33

4.3 Experimental results for selected ML/DL algorithms with CICIDS2017 dataset. 38

4.4 The EIR(%) of the IDSs for adversarial samples of DoS Hulk, DDoS, DoS GoldenEye, and PortScan generated by WGAN. . . 41

4.5 The EIR(%) of the IDSs for different types of adversarial samples generated by the WGAN and CWGAN. 51

List of Figures

2.1 A schematic illustration of a conditional discriminator and a conditional generator 12

3.1 A schematic illustration of a Multilayer Perceptron Neural Network. 18

3.2 A schematic illustration of a one-dimensional Convolutional Neural Network. 19

3.3 A schematic illustration of a Recurrent Neural Network. . . . 21

3.4 A schematic illustration of the proposed WGAN-based framework. 22

3.5 An example of embedded vectors for class labels 26

3.6 A schematic illustration of the CWGAN-based framework . . . 26

4.1 An illustration of the network infrastructures used to generate the CICIDS2017 dataset 30

4.2 Evolution of average loss versus epoch for MLP, RNN, and CNN. 35

4.3 Evolution of test accuracy versus epoch for MLP, RNN, and CNN. 36

4.4 Confusion matrix for the selected ML/DL algorithms. 37

4.5 The detection rate of the selected ML/DL algorithms with the original and adversarial datasets for DoS Hulk. 39

4.6 The detection rate of the selected ML/DL algorithms with the original and adversarial dataset for DDoS. 40

4.7 The detection rate of the selected ML/DL algorithms with the original and adversarial dataset for DoS GoldenEye. 41

4.8 The detection rate of the selected ML/DL algorithms with the original and adversarial datasets for PortScan. 42

4.9 The detection rate of the Naive Bayes algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 43

4.10 The detection rate (accuracy) of the Logistic Regression algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 44

4.11 The detection rate of the Decision Tree algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 45

4.12 The detection rate of the Random Forest algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 46

4.13 The detection rate of the Gradient Boosting algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 47

4.14 The detection rate of the Multilayer Perceptron with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 48

4.15 The detection rate of the Recurrent Neural Network with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 49

4.16 The detection rate of the Convolutional Neural Network with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN 50

5.1 A schematic illustration of a Long Short Memory Term network used as a generator 54

5.2 A schematic illustration of a variational Autoencoder 55

5.3 A schematic illustration of the Earth Mover Distance between two one-dimensional distribution 55

5.4 The absolute difference between correlation matrices computed on real and synthetic datasets 56

5.5 A schematic illustration of the federated learning technique. . 58

Abstract

Intrusion detection systems (IDSs) detect malicious activities in network flows and is essential for modern communication networks. Machine learning (ML) and deep learning (DL) have been employed to construct IDSs. However, the reliability of ML/DL-based IDSs is questionable under adversarial attacks. We propose a framework based on Wasserstein generative adversarial networks (WGANs) to generate adversarial traffic to evade ML/DL-based IDSs. The proposed framework involves restricted modification operations and the output of the framework is carefully regulated, preserving the functionality of the malicious attack. We also present a variant of the proposed framework based on conditional WGANs. The variant framework simplifies the training procedure without losing attack capability. Eight ML/DL-based IDSs are constructed, and their robustness against adversarial attacks is tested using the frameworks. The results show that the framework and its variant can generate adversaries effectively, and the Convolutional Neural Network has the best robustness under adversarial attacks.

Acknowledgements

Thank you, Dr. Qiang Ye, for your time and your guidance. Thank you, Baorui Jia, for helping me get through the difficulties here in Halifax. Thank you to all my colleagues and teammates who devoted a great deal of their time and energy to me.

Chapter 1

Introduction

1.1 Intrusion Detection

Network intrusion detection is a significant task for cyber security. The key purpose of a network intrusion detection is to monitor network traffic to detect and report cyber security problems such as malware, denial of service (DoS), and port scans. An intrusion detection system (IDS) is usually a computer software that has the required detection functionalities. With the rapid growth in network traffic and cyber attacks on the Internet, IDSs have become essential tools to detect and defend against network attacks. The IDSs aim to monitor the network flows and identify malicious traffic by analyzing the features extracted from these flows. The IDSs are expected to classify a traffic flow as legitimate or malicious with high accuracy and a low false positive rate.

The first-generation IDSs are signature-based [40]. The signatures are usually characteristic features of known types of attacks and can be found by analyzing detected attack traffic flows. The signature-based IDSs performed their detection tasks based on such pre-defined signatures. In other words, any monitored network traffic flows will be treated as malicious if the pre-defined signatures are spotted in them. However, the signature-based IDSs require frequent updates to deal with newly emerged attacks as they are built upon the pre-known attack information. Any out-of-date information can prevent signature-based IDSs from detecting such new attacks (i.e., zero-day attacks).

Anomaly-based IDSs attempt to learn the normal behaviours of monitored networks and classify monitored system activities as normal or anomalous. Any observed network activities that fall out of normal system operations will be treated as anomalous/malicious and trigger the alert. Anomaly-based IDSs perform the detection task based on heuristics or rules instead of traffic patterns or pre-defined signatures, which is opposed to signature-based IDSs. As anomaly-based IDSs do not rely on pre-known knowledge of cyber attacks, they have great potential on detecting new types of attacks in a system and also do not require frequent maintenance. The major shortcoming of anomaly-based IDSs is that it usually requires a large amount of data for them to thoroughly learn the normal behaviors of the monitored network.

Hybrid IDSs were developed to overcome the drawbacks of signature-based IDSs and anomaly-based IDSs. A hybrid IDS is usually built up by incorporating a signature-based IDS and an anomaly-based IDS into a unifying system. In the system, two IDSs can be deployed in parallel, and the final prediction of the hybrid system is decided by weighing the result from each IDS. The weighting strategy can be updated periodically based on the performance of the hybrid system. A hybrid IDS generally outperforms the sub-IDSs that compose it. A well-established hybrid IDS is good at detecting both known and unknown cyber-attacks and meanwhile can avoid a high false positive rate. Though hybrid IDSs show obvious advantages, compared to signature-based IDSs and anomaly-based IDSs, it takes a lot more effort to construct it. It is a challenging task to get IDSs with different technologies to inter-operate successfully and efficiently. Therefore, anomaly-based IDSs are still the most widely studied ones in the community.

Due to recent development and growths in machine learning (ML) and deep learning (DL) techniques, a large number of different ML/DL algorithms have been employed to address various types of classification tasks. They have shown high efficiency and outstanding performance in categorizing and classification within a short amount of time. A similar trend has been found in the cybersecurity field such as intrusion and malware detection. As a typical classification task, ML/DL algorithms have been used to build up anomaly-based IDSs due to their simplicity, efficiency, and outstanding performance [53]. However, these astonishing ML/DL algorithms are not flawless. It has been proved in a recent work that the ML/DL algorithms expose vulnerability when subjected to adversarial attacks, in which well-designed adversarial inputs lead to the misclassification of ten different ML/DL algorithms [9]. A fundamental assumption in ML/DL-based systems is that the distribution followed by the training data set will also be encountered at inference time, which may not be the case in the real world. Adversarial attacks utilize this drawback and employ search and optimization algorithms to generate adversarial samples (e.g., by adding perturbations on original samples), which can compromise the performance of the ML/DL algorithms.

1.2 Adversarial Attacks

Adversarial attack refers to a type of attack methodology, in which perturbations are added to original samples to generate adversarial samples to deceive trained ML/DL algorithms, leading to a high probability of misclassification. Adversarial attacks have been widely investigated in the image processing field. For example, after adding perturbations on image samples, generated new images can flip the results of ML/DL algorithms and meanwhile retain their visualization for human eyes [37]. In recent years, adversarial attacks have been successfully used on tabular data [10] as well, and the results are also promising compared to attack results on image data sets.

Many adversarial attack methods have to know detailed information about an ML/DL scheme (i.e., algorithms, architectures, parameters, etc.) to launch attack by generating adversarial samples. Such attack methods are referred to as “white-box” adversarial methods. For instance, the well-known Fast Gradient Signed Method (FGSM) makes use of gradients of an artificial neural network to create adversarial samples [21]. The FGSM examines the gradients of the loss with respect to the input sample to create a new sample that maximizes the targeted loss function. Without knowing the architecture and parameters of the artificial neural network, the gradients of the loss can not be calculated and monitored (i.e., doing a back-propagation), and hence, the FGSM is not able to be used for launching adversarial attacks under such scenarios. Hence, the “white-box” methods are usually adopted for theoretical study and provide useful knowledge and benchmark information regarding the adversarial attacks on IDSs. In real-world cyber-attack scenarios, the detailed information of the targeted ML/DL IDS is usually unknown to attackers, which is like a “black box”. Therefore, the adversarial attack methods which can deceive ML/DL algorithms without knowing their detailed information are usually called ‘black-box’ adversarial methods. Black-box adversarial methods are more useful and practical when studying real-world cyber-attack scenarios.

Generative adversarial networks (GANs) [20] belong to a family of deep generative algorithms and are good potential candidates for launching black-box adversarial attacks. The core idea of GANs is to have two players, namely a generator and a discriminator, interactively play against each other. The task of the generator is to generate samples that can best approximate the training set, whereas the goal of the discriminator is to distinguish the generated samples from the original training samples. In addition, by using the predicted labels from IDSs (ML/DL-based) as benchmarks, the discriminator can further mimic the performance of IDSs without knowing their detailed information. Therefore, the generator can finally be trained to deceive the IDSs by getting feedback from the discriminator. This is how black-box adversarial attacks can be launched by GANs. Wasserstein generative adversarial networks (WGANs) are an improved version of the original (vanilla) GAN, which overcomes the shortcomings of the original GANs, such as mode collapse and difficulty of convergence, by introducing the Wasserstein loss function [57]. Another improved derivative of GANs is called conditional generative adversarial networks (CGANs) [43]. The CGANs have the capability of generating adversarial samples of multiple classes simultaneously by introducing conditions or label embeddings. CGANs significantly improve the efficiency of GANs as there is no need to retrain GANs (both generator and discriminator) for different classes of input samples.

1.3 Overview of Wasserstein GAN Based Framework

In the present study, we first build up a WGAN-based framework to compromise the performance of the ML/DL-based IDSs by generating adversarial samples. Eight ML/DL algorithms are selected for the present study, namely, Naive Bayes, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, Multi-layer Perceptron, Recurrent Neural Network and Convolutional Neural Network. Instead of building up and evaluating the framework in a real-time network system, we elect to use the recently published CICIDS2017 [49] dataset for the sake of efficiency and effectiveness. Four majority types of attack records from the dataset, DDoS, Dos Hulk, DoS GodlenEye, and Portscan are used for the present study. Furthermore, the designed WGAN framework has restricted modification mechanisms and output regulations which enable our WGAN to compromise the performance of ML/DL-based IDSs and meanwhile preserve functionalities of attack records. The performance of the proposed framework is tested using four different attack traffic records on eight widely used IDSs. Our results indicate that the generated adversarial samples of attack records by the WGAN can significantly reduce the detection accuracy of the studied IDSs. Thereafter, we further improve the proposed WGAN framework and alternate it into a CWGAN framework. The designed CWGAN framework can be trained and generate adversarial samples of four different attack types simultaneously. The performance of the CWGAN framework is generally consistent with the WGAN framework, which reflects the success of the CWGAN method.

1.4 Thesis Outline

The rest of the thesis is organized in the following manner. In the next chapter, we present the related works regarding ML/DL-based IDSs, generative adversarial network (GAN) and two improved versions of the vanilla GAN (e.g., WGAN and CGAN). We also discuss published literature related to adversarial attacks on IDSs in Chapter II, including both white-box and black-box adversarial attack methods. Chapter III introduces the details of the WGAN-based attack methodology, including the fundamental framework, the architecture of the generator and discriminator, and the details of the adopted algorithm. In addition, it is described in this chapter how we alternate our WGAN-based framework into a more advanced CWGAN-based one. Chapter IV first provides detailed information on how we test the proposed framework, including the description of the dataset, how we perform the data preprocessing, the selected features for modification, and the experimental setup and evaluation metrics. After this information is provided, the experimental results for both WGAN and CWGAN-based frameworks are shown in Chapter IV. In the last chapter, the thesis is concluded with the current milestone, and several further research directions are proposed and discussed.

Chapter 2

Related Work

In this chapter, we first discuss the published work related to IDSs, including both signature-based and anomaly-based ones. Then we introduce the background knowledge for generative adversarial network (GAN) and their improved derivatives. Lastly, the published literature regarding cybersecurity-related adversarial attacks will be reviewed. Note that as IDSs, ML/DL algorithms, and adversarial attacks are all popular research topics, more than thousands of related research articles have been published so far. Thus, it is unfeasible to review all of them and only representative ones will be introduced.

2.1 Intrusion Detection Systems: From Signature to Deep Learning

IDSs are software applications that monitor network traffic flows to recognize harmful ones. It then issues alerts to administrators if malicious traffic flows in the system are determined and detected. IDSs can be divided into two types based on the detection strategy used: Signature-based IDSs and Anomaly-based IDSs.

The signature-based IDS has a long history and the first published article can date back to the late 1980s by D.E. Denning [13]. In Denning's work, a real-time signature-based IDS was developed using abnormal patterns obtained by monitoring the system's records. The designed IDS is independent of any particular system and application environment. Denning's work was used as a reference for many later publications. Since then, there are a large number of researchers that published on signature-based IDSs, such as Ilgun [23], Yang et al. [60], and Uddin et al. [54], among many others. Due to the development of ML/DL algorithms and the obvious shortcomings of signature-based IDSs (i.e., need frequent updates, and are fragile when subjected to zero-day attacks), the study of signature-based IDSs gradually decreased in the last two decades. However, as the most classical type of IDS, it is still used and studied by some researchers these days. Ioulianou et al. [24] developed a signature-based IDS for the Internet of Things (IoT). The IDS was tested with Denial of Service (DoS) attacks using the Cooja simulator. Also, Li et al. [32] designed a signature-based IDS based on collaborative blockchain, and the proposed system performed well in different environments and adversarial scenarios. The contribution of signature-based IDSs to cyber-security is not negligible.

Anomaly-based IDSs have become more popular in the last decade due to their outstanding performance and unique advantages [19]. Thanks to the remarkable capability and continuous development of ML/DL algorithms, they have been widely adopted to build anomaly-based IDSs and have achieved initial potential results [53]. In early research, machine learning algorithms, either supervised or unsupervised ones, are usually used by researchers due to their high availability and low computing complexity. For instance, Eskin et al. [18] developed a geometric framework for unsupervised anomaly-based detection. In the geometric framework, data features are mapped to a feature space (usually a vector space) using pre-defined kernels, and anomalies are detected by determining which data points lie in spare regions of the feature space. The proposed geometric framework was tested using the KDD CUP 1999 [51] dataset with three unsupervised machine learning algorithms: clustering, K-nearest neighbour, and one-class support vector machine. With their framework, the detection rate can reach up to 98% with a false positive rate that is less than 10%. Jiang et al. [25] developed a cluster-based method for intrusion detection using improved nearest-neighbour algorithms. Their method outperformed the framework proposed by Eskin et al. [18]. In addition, they analyzed the time complexity of the method and showed their algorithm is linear with the size of the dataset and the number of attributes. These anomaly-based IDSs with unsupervised learning algorithms are very useful when handling unlabeled data sets.

When it comes to anomaly-based IDSs using supervised learning algorithms, Yang and Li [33] developed an active learning framework for supervised network intrusion detection using the Traductive Confidence Machines for K-Nearest Neighbors (TCM-KNN) algorithm. They also adopted KDD CUP 1999 data set to test the framework and the detection rate of the TCM-KNN algorithms can reach up to 99% with a false positive rate that was smaller than 0.5%. Meng [42] examined the performance of the Support Vector Machine (SVM) and Decision Tree (DT) on network intrusion detection. Instead of using the KDD CUP 1999 data set as a whole, Ment reported the performance of ML algorithms on each subtype of the attack records in the dataset, namely, Probe, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L). The detection rate of the SVM and DT algorithms is higher than 95% except for the U2R attack. The poor performance of the ML algorithms on the U2R attack is due to the lack of sufficient U2R records in the data set. Choudhury et al. [11] compared the performance of a series of ML algorithms, such as Naïve Bayesian (NB), Logistic Regression (LR), and Random Forest (RF), on network intrusion detection using NSL-KDD dataset [51], which is an improved version of the KDD CUP 1999. Their investigation shows that most of the ML algorithms can provide promised results on anomaly-based intrusion detection.

In recent years, with the rapid development of computing hardware, deep learning algorithms have grown rapidly and started to be used in the intrusion detection domain. Ding and Zhai [15] developed a Convolutional Neural Network (CNN) for anomaly-based intrusion detection. In their CNN, three one-dimension convolutional layers were employed to extract features from original data records, and subsequently, the extracted features are sent to several dense layers for classification. The output layer of the CNN is a softmax function considering multiple attack classes in the NSL-KDD Dataset. The designed CNN was compared with four other ML/DL algorithms, namely, Random Forest, Support Vector Machine, Artificial Neural Network, and Long Short Term Memory. Their results showed that the designed CNN outperformed other algorithms on both the detection accuracy and the false positive rate. Tang et al. [50] developed a deep Recurrent Neural Network (RNN) for an anomaly-based intrusion detection system used in software-defined networks (SDNs). Tang et al. built up the RNN with Gate Recurrent Units (GRUs) to form a GRU-RNN. The proposed GRU-RNN outperformed the vanilla RNN and regular deep neural networks (DNN), and the detection accuracy reached up to 89% for the NSL-KDD dataset with only six raw features. In addition, Tang et al. performed analysis using Area under the Receiver Operating Characteristics (AUROC) curve, and the AUROC of GRU-RNN is equal to 0.90, which is higher than 0.80 and 0.50 for DNN and vanilla RNN, respectively. These DL algorithms with ML algorithms together improve the accuracy and simplify the intrusion detection problem in network systems.

2.2 Generative Adversarial Network

Generative adversarial network (GAN), which is first proposed by Goodfellow et al. [20], is a type of deep learning-based generative algorithm. GAN aims to produce adversarial samples by using a given dataset, so that generated samples are as similar to input ones. There are two deep learning-based sub-components which compete against each other in a GAN, namely, the generator G and the discriminator D. The task of the generator is to create adversarial samples which are expected to be as real as input data. Whereas the discriminator, as the adversary of the generator, tries to distinguish the adversarial samples (fake data) from the input data (real data) [8]. In the training process of a GAN, the two sub-components are trained simultaneously and play against each other based on the zero-sum game theory. The generator wins the game If the output of the generator can successfully deceive the discriminator, and loses the game when the discriminator identifies the produced fake samples. Both of the components update their parameters based on the competing results of the current round and intend to improve their operation in the next round of the game (training). Ideally, a GAN reaches its convergence state (e.g., the zero-sum game ends) when no sub-component can win over the other one. In the convergence state, the generator can produce adversarial samples that result in uncertain classification (fifty-fifth results like flip the coins) of the discriminator. Subsequently, the generator can operate separately without the discriminator to generate adversarial samples.

Based on Goodfellow et al. [20], the loss function of the generator is given by:

$$\underset{\theta}{min}\, L_G(\theta, \phi) = \mathbb{E}_{z \in p_Z(z)}[\log(1 - D_\phi(G_\theta(z)))] \tag{2.1}$$

and the loss function of the discriminator is given as:

$$\underset{\phi}{max}\, L_D(\theta, \phi) = \mathbb{E}_{x \in p_X(x)}[\log D_\phi(x)] + \mathbb{E}_{z \in p_Z(z)}[\log(1 - D_\phi(G_\theta(z)))] \tag{2.2}$$

In Eqs. (2.1) and (2.2), the θ and ϕ represent the parameters of the generator and discriminator, respectively, which are supposed to be optimized through training. The Z is the latent space containing noise vectors z (e.g., $z \in Z$) for adversarial sample generation. The $P_Z(z)$ is the distribution over the latent space. In addition, the X stands for the data space that includes all input samples (e.g., $x \in X$), and $P_X(x)$ denotes the distribution of the real samples in the data space. It is noticed in Eqs.

(2.1) and (2.2) that both generator G and discriminator D rely on their counterpart's feedback or output to update their parameters. Therefore, the two equations can be merged into a general loss function that describes the optimization process of the vanilla GAN, given by:

$$\underset{\theta}{min}\,\underset{\phi}{max}\, L_{GAN}(\theta,\phi) = \mathbb{E}_{x\in p_X(x)}[\log D_\theta(x)] + \mathbb{E}_{z\in p_Z(z)}[\log(1 - D_\phi(G_\theta(z)))] \tag{2.3}$$

Eq. (2.3) indicates that the optimization of a GAN is a mini-max task between the generator G and the discriminator D. For the vanilla GAN (using Eq. (2.3)), the training is performed by mapping a pre-defined latent space ($z \in Z$) to a space of real/input samples ($x \in X$) for distinguishing. On the one hand, the G has to generate a distribution for $z \in p_Z\left(z\right)$ corresponding to $x \in X$. After all the real/input samples ($x \in X$) running through the GAN (e.g., after one training epoch), the distribution of the generated samples given by G can be computed and evaluated (i.e., comparing with input samples). Note that the distribution $p_Z(z)$ of the latent space is usually designed with a standard Gaussian or uniform $[0,1]$ distribution. On the other hand, the discriminator D works as an adversary of the generator G to distinguish the input samples ($x \in X$) from the generated ones $G_\theta\left(z\right)$ produced by the generator G. The discriminator D is trained to give predictions close to 1 for the real/input samples, but nearly 0 for the case of the fake/generated samples (like an output of an ideal binary classifier). This means that to optimize the loss function the discriminator intends to maximize the term $D_\phi(x)$ but minimize the term $D_\phi(G_\theta(z))$. Note that the task of minimizing the term $D_\phi(G_\theta(z))$ is equivalent to maximizing the term $(1-D_\phi(G_\theta(z))$. Furthermore, to fool the discriminator D by inducing labels from the input samples, the term $D_\phi(G_\theta(z))$ needs to increase to 1 to maximize $D_\phi(G_\theta(z))$, which, in other words, is the same as minimizing $(1 - D_\phi(G_\theta(z)))$.

In a word, the generator and discriminator are in opposition during the training, while D tries to maximize the loss function (e.g., distinct all fake samples from the real data), G tries to minimize the loss function (e.g., generate samples as real as possible to completely fool D). The D and G stop updating their parameters when the GAN reaches the equilibrium state between the two sub-components, which is referred to as the Nash equilibrium.

2.2.1 Wasserstein GAN

Though GAN has shown great success in many fields such as image processing [22] and tabular data generation [59], it is not an easy task to optimize a GAN [57]. There are several reported problems which make the training process of GANs slow and unstable. One most common failure situations in GAN training is called mode collapse [52]. When optimizing the discriminator and generator, one possibility is that the generator becomes much stronger and can completely fool the discriminator with the adversarial samples. Under such situations, the discriminator is no longer able to provide any useful feedback to the generator (see Equation 2.1), and hence, the generator stops updating its parameters to capture the complicated data distribution of the original input, resulting in extremely low diversity of the generated adversarial samples. On the contrary, sometimes the discriminator can also overwhelmingly dominate the training process and distinguish all the produced samples (fake) from the input ones (real). In these cases, the gradient descent value of the loss function (the feedback to the generator) vanishes, and the generator can learn nothing from the discriminator to improve its performance on adversarial sample generations. Subsequently, the training process gets stuck, and the game will never end. In a word, it is very tough to train a GAN: the GAN may never converge and mode collapse occurs frequently.

To overcome the shortcomings of the vanilla (original) GAN, Arjovsky et al. designed Wasserstein GAN (WGAN) to improve the GAN training process [4]. In a WGAN, the original log-based loss function is replaced with a new loss function derived from the Wasserstein distance. Also, instead of classifying the data as real (e.g., input data) or fake (e.g., generated data), the new discriminator provides a score to evaluate the quality of the data. This score indicates how realistic the generated data is with respect to the input data distribution. The authors named the new discriminator critic with the responsibility of calculating the Wasserstein metric between the input data distribution and the adversarial data distribution. The loss function of the Wasserstein GAN will be introduced in the next section with the details of the proposed WGAN-based framework.

2.2.2 Conditional GAN

Being first introduced by Mirza and Osindero [43], the conditional generative adversarial network (CGAN) is the conditional version of the vanilla GAN. Mirza and Osindero delineate the problem using the famous MINST digit image dataset [29]. With the original GAN, they are not able to control what specific images (handwritten digits) the generator will produce. In other words, there is no way that they request the generator to provide a particular digit number (e.g., 0 to 9). The designed CGAN successfully solves this problem by adding input conditions to both the generator and discriminator. A schematic illustration of a conditional generator and a conditional discriminator is shown in Figure 2.1. The vector x in Figure 2.1 represents the features of the input data, and the vector y denotes the vector containing condition information. The input conditions y are usually based on but are not limited to class labels (i.e., one-hot encoded vectors or word embedding vectors). These information vectors are usually concatenated with the original input data features x, and then they are fed to the generator and discriminator for training.

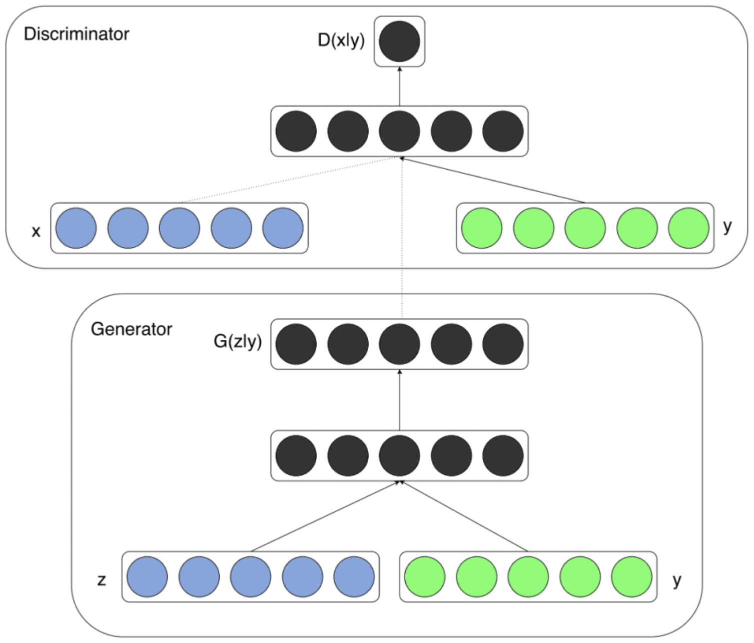


Figure 2.1: A schematic illustration of a conditional discriminator and a conditional generator [43].

Mirza and Osindero show that for a well-optimized CGAN the generator can successfully produce MINST digit samples based on the provided conditional class labels. In addition, they found out that the CGAN could be used to learn a multi-modal distribution, and provided preliminary results of an application to image tagging in which it is demonstrated how the CGAN method can generate descriptive tags which are not part of labels from the training data set.

Note that adding condition information to the generator and discriminator of a GAN does not alternate its loss functions. Hence, such information can not just be added to the vanilla GAN but also to any kind of GAN derivatives. For instance, Zheng et al. [62] developed a conditional Wasserstein generative adversarial network (CWGAN) to improve the classification of imbalanced data with generated data samples. Lin et al. [34] designed a conditional dual GAN for image-to-image translation. Both of the aforementioned two works provide promised results.

2.3 Adversarial Attacks against Intrusion Detection Systems

Despite certain successes, ML/DL-based IDS methods have been proven to be vulnerable to adversarial attacks. In recent years, investigations of adversarial attacks on IDSs started to emerge. Adversarial attacks can be categorized into two types, namely, white-box adversarial attacks and black-box adversarial attacks, based on the attackers' knowledge regarding the targeted IDS. Regarding white-box attacks, the attacker has access to all the detailed information of the IDS (i.e., algorithms, architecture, and internal parameters), while in black-box attacks, the attacker has no access to such information at all. In real-world scenarios of network attacks, the information of IDSs is usually unknown to attackers. Thus, though white-box attacks can be more efficient as attackers can launch the attack by analyzing the weakness of the targeted IDS, black-box attack methodologies are more useful and practical in real-world attacks.

2.3.1 White-box Adversarial Attacks

Rigaki and Elragel [47] employed Jacobian-based Saliency Map Attack (JSMA) and Fast Gradient Sign Methods (FGSM) algorithms to craft adversarial samples with the famous NSL-KDD dataset [1]. The generated adversarial samples successfully decreased the detection rate (about 20 percents) of several ML-based IDSs, including Decision Tree, Random Forest, Linear SVM, Voting ensemble, and MLP. Wang [56] further leveraged multiple adversarial algorithms (including JSMA, FGSM, DeepFool and CW) to generate adversarial samples. Though the results given by Wang [56] are more promising by examining the algorithm detection rate with adversarial samples, they altered all the features of traffic records from the dataset during the sample generation, which surely impact the original functions of the network traffic. The aforementioned two studies assumed that the attackers have all knowledge about the targeted IDSs, which belong to white-box attacks.

2.3.2 Black-box Adversarial Attacks Using GAN

In a real-world cyber attack, the detailed information of IDSs (i.e., detection algorithm and value of parameters) is usually unknown to attackers. Generative adversarial network (GAN) allows attackers to launch black-box attacks (e.g., without knowing all the information of the targeted IDS) by mimicking the performance of the targeted IDS using a discriminator. Lin et el. [35] proposed the IDSGAN framework to generate traffic records for adversarial attack on IDSs, which is capable of misleading the classification of several pre-trained ML/DL-based IDSs. They employed a WGAN as the backbone of their system and evaluated the framework using the famous NSL-KDD dataset. However, another recent study by Usama et al. [55] indicated that the IDSGAN framework has changed several functional features of attack traffic records, and hence, the generated adversarial records may no longer preserve the original functionalities. Instead of using a WGAN, Usama et al. adopt a basic (vanilla) GAN to build up their system. In addition, they divide the traffic features into functional ones and non-functional ones based on different attack types, and the modification only occurs on non-functional features when generating adversarial samples, aiming to preserve the traffic functionalities. Phan et al. [17] developed a framework similar to IDSGAN and performed investigations using a relatively new dataset CICIDS2017 [49], which contains fourteen different types of attack records. They performed adversarial attacks on nine ML/DL-based IDSs using the DDoS malicious records in the CICIDS2017. In their results, the detection rate of all studied IDSs decreased to nearly zero for generated adversarial records. However, Phan et al. did not add regulations on the output layer of the generator of the WGAN, which may cause unreasonable values in the generated adversarial samples (i.e., extremely large and negative values).

Chapter 3

Wasserstein GAN Based Adversarial Attacks

In this chapter, we delineate our methodology for adversarial attacks against IDSs. The framework proposed herein is based on previous studies by Lin et al. [35], Usama et al. [55], and Phan et al. [17] on the adversarial attack against IDSs. In particular, compared with the latest publication by Phan et al. [17], there are four major improvements: First, four different types of attack records are considered instead of the DDoS solely; second, we provide a more complete list of functional features, and these functional features are fixed during framework training; third, the output of the generator is highly regulated to ensure all generated feature values are within reasonable ranges; last but not least important, we design a variant of the framework based on conditional GAN, with which the training procedure is significantly simplified but without reducing the framework capability.

In the first section of this chapter, we discuss the details of the selected ML/DL algorithms, such as algorithm hyperparameters and neural network architectures. Then, we present the details of the WGAN-based framework for adversarial attack, including the scheme, the architecture of the generator and the discriminator, and the flow of the designed algorithms. Lastly, we show how the proposed WGAN-based framework can be modified into a conditional one using embedded class labels. The modified architecture of the conditional GAN and the related flow of the algorithm will also be given.

3.1 Selected Schemes for IDS

3.1.1 Machine Learning

Five widely used machine learning algorithms were selected in the present study for constructing the IDSs, namely, Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Gradient Boosting (GB). The NB and LR are relatively simple algorithms, but they are widely used to get benchmarks for other algorithms. The DT is a kind of tree-based ML algorithm and has shown good performance in classification tasks of IDSs [28]. The RF and GB algorithms belong to tree-based ensemble learners, and they are based on bagging and boosting strategies, respectively. The RF and GB algorithms could always provide promised results on challenge classification tasks [46] [41].

3.1.2 Multilayer Perceptron

The Multilayer Perceptron (MLP) is one of the most widely used deep-learning algorithms for classifications. A schematic illustration of a general MLP architecture is shown in Fig. 3.1. There are three types of layers in a general MLP architecture: input layer, hidden layer, and output layer. The number of neurons in the input layer depends on the number of features of the input data. The number of neurons in the output layer depends on classification tasks. Only one neuron is required in the output layer for binary classification, and for multi-class classification tasks, the number of neurons in the output layer is equal to the number of class labels in the dataset. There is no rule of thumb for the number of hidden layers and the number of neurons in each hidden layer. They have to be adjusted carefully case by case as too many neurons and hidden layers may cause overfitting whereas too less neurons and hidden layers may affect the capability of a neural network.

The architecture of the MLP used in the present study is described in Table 3.1. We used three hidden layers with ReLU activation functions to ensure the neural network has sufficient non-linearity and capability. The output layer has a Sigmoid activation function as it is for a binary classification task. The number of neurons in each layers depends on the dimension of the input features, which is reasonable.

3.1.3 Convolutional Neural Network

Though the Convolutional Neural Network (CNN) was originally developed for image processing, it has been proven to have a good performance in manipulating tabular data. A schematic illustration of a one-dimensional CNN is shown in Fig. 3.2. Similar to image processing, a one-dimensional convolutional layer is responsible to extract features from the input data features, and then a pooling layer is adopted

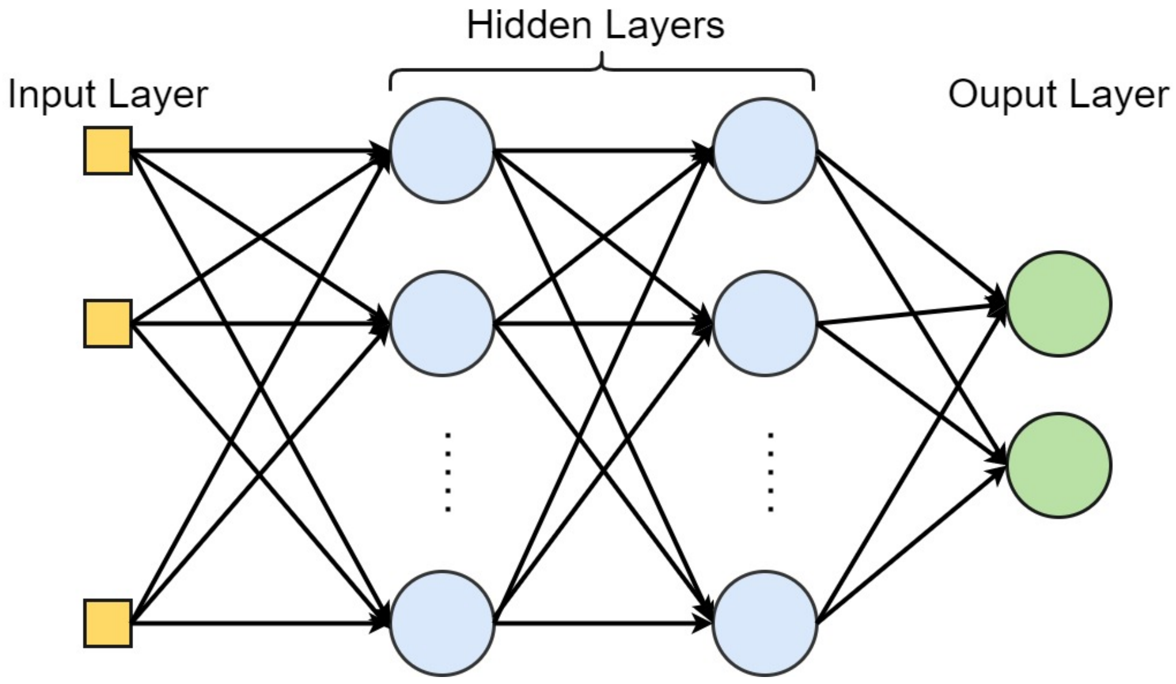


Figure 3.1: A schematic illustration of a Multilayer Perceptron Neural Network.

to find the characteristic features. Subsequently, the resulting feature channels are sent to a fully-connected layer for classification. Multiple convolution layers and fully-connected layers can be used to increase the non-linearity (capability). Note that though the feature extraction function of convolutional layers can significantly improve the capability of the neural network, training such a convolutional neural network is time-consuming. We do not consider the computational feasibility in the present study, but it is a matter of fact that it is very challenging to deploy a large CNN on a mobile device.

Table 3.1: Description of the architecture of the Multilayer Perceptron for the IDS.

Operation	**Input Units**	**Output Units**	**Non-linearity**
Linear	input_dim	input_dim*2	ReLU
Linear	input_dim*2	input_dim*2	ReLU
Linear	input_dim*2	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	1	Sigmoid

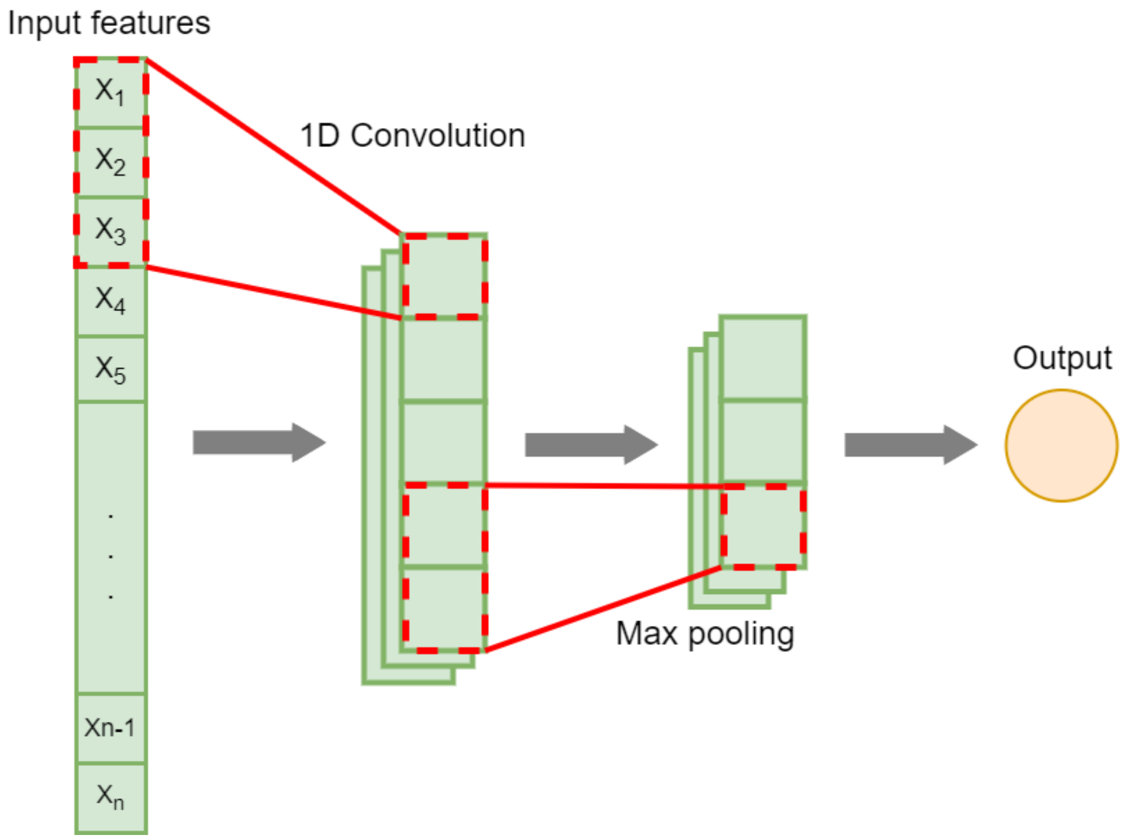


Figure 3.2: A schematic illustration of a one-dimensional Convolutional Neural Network.

The Convolutional Neural Network (CNN) used in the present study is described in Table 3.2. Two one-dimensional convolutional layers (ReLU non-linearity) with max-pooling functions are employed for feature extraction. The extracted features are then flattened before feeding into two fully connected (linear) layers for classification. Note that in image classification tasks the scale of the CNN is much larger than the one adopted herein [27]. Large-scale CNN were also tried for the present study. However, the authors did not see obvious improvement after using a large-scale CNN, and the training speed significantly slowed down. Thus, the authors kept the proposed CNN in Table 3.2.

3.1.4 Recurrent Neural Network

The Recurrent Neural Network (RNN) is usually to process time sequence data. In the present study, the input traffic flows are not time sequences, and hence we treat

Table 3.2: Description of the architecture of the Convolutional Neural Network for the IDS.

Operation	**Parameter**	**Non-linearity**
Conv1d	(1, 4, kernel = 4)	ReLU
Maxpool1d	(4)	-
Conv1d	(4, 8, kernel = 4)	ReLU
Maxpool1d	(2)	-
Flatten	-	-
Linear	(64, 32)	ReLU
Linear	(32, 1)	Sigmoid

them like a pseudo time sequence with only a one-time step. A schematic illustration of using a RNN for IDSs is shown in Fig. 3.3. The RNN cell is initialized with some hidden states. First, the preprocessed input features are fed into the RNN cell. With the hidden states from the previous iteration and the input feature, the RNN cell will update the hidden states, and then the updated hidden states will be sent to fully-connected layers for classification.

Table 3.3: Description of the architecture of the Recurrent Neural Network for the IDS.

Operation	**Parameter**	**Non-linearity**
RNN	(input_dim, 60)	-
Linear	(60, 30)	ReLU
Linear	(30, 1)	Sigmoid

The architecture of the RNN used in this thesis is described in Table 3.3. We adopted a single recurrent layer, and as the input data is not a sequence mode, only one RNN cell was used in the layer. The details of the RNN cell are fundamental knowledge in deep learning, and will not be discussed herein. It is proved in a later section that the RNN can provide promised results on classification.

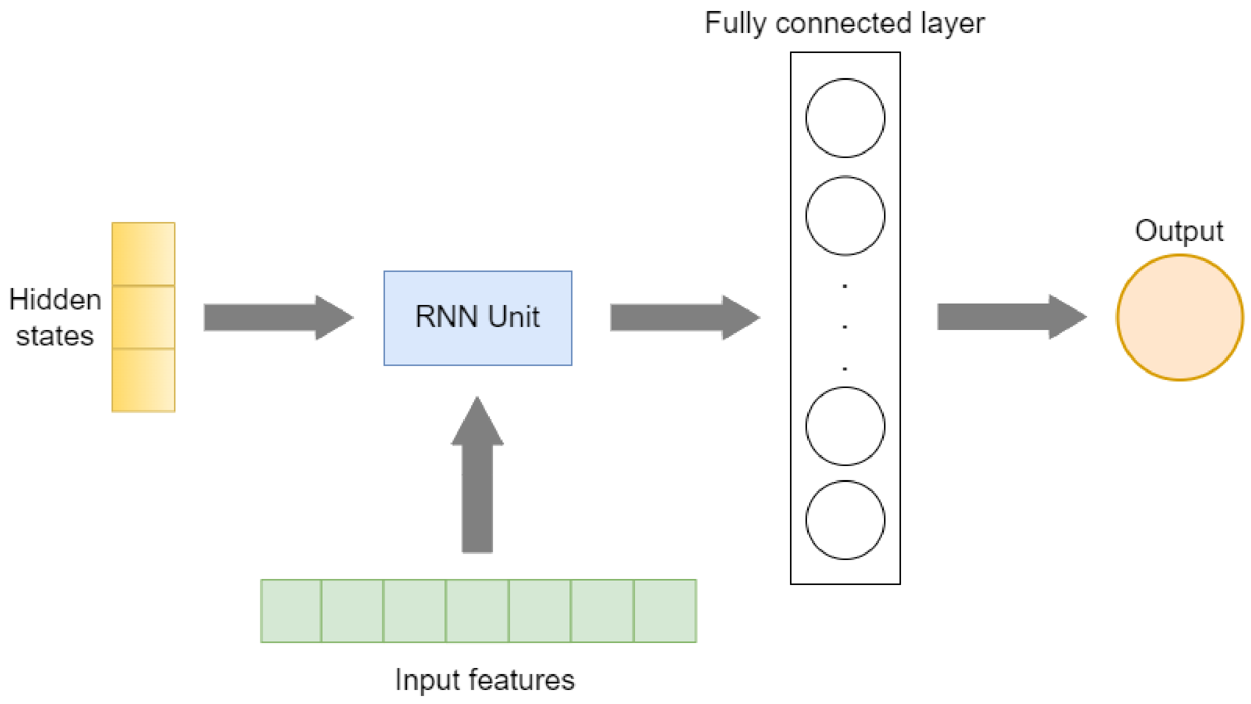


Figure 3.3: A schematic illustration of a Recurrent Neural Network.

3.2 Wasserstein GAN Based Framework

The schematic illustration of the proposed WGAN Framework is delineated in Figure 3.4. There are three major components in the framework, namely, a Generator, a Discriminator, and an IDS. The IDS is pre-trained and its parameters are fixed during the whole process. Also, in a real-world network attack, the detailed structure of the IDS is usually unknown and is like a “black box” for attackers. To evade the detection of the ”black-box” IDSs, the generator modifies features of the original attack traffic records to generate adversarial attack traffic records. The modified attack traffic records along with normal traffic records are then fed to both the discriminator and the IDS. The discriminator is trained to mimic the performance of the black-box IDS, and this is realized by obtaining the prediction outcomes (labels) of the input traffic records from the IDS. Subsequently, the discriminator provides feedback to the generator, and the generator updates itself based on such feedback.

Though the proposed WGAN framework aims to modify the attack traffic records to evade the detection of the IDS, such modifications should retain the functionality of the original attack traffic so that the adversarial attack traffic can still launch

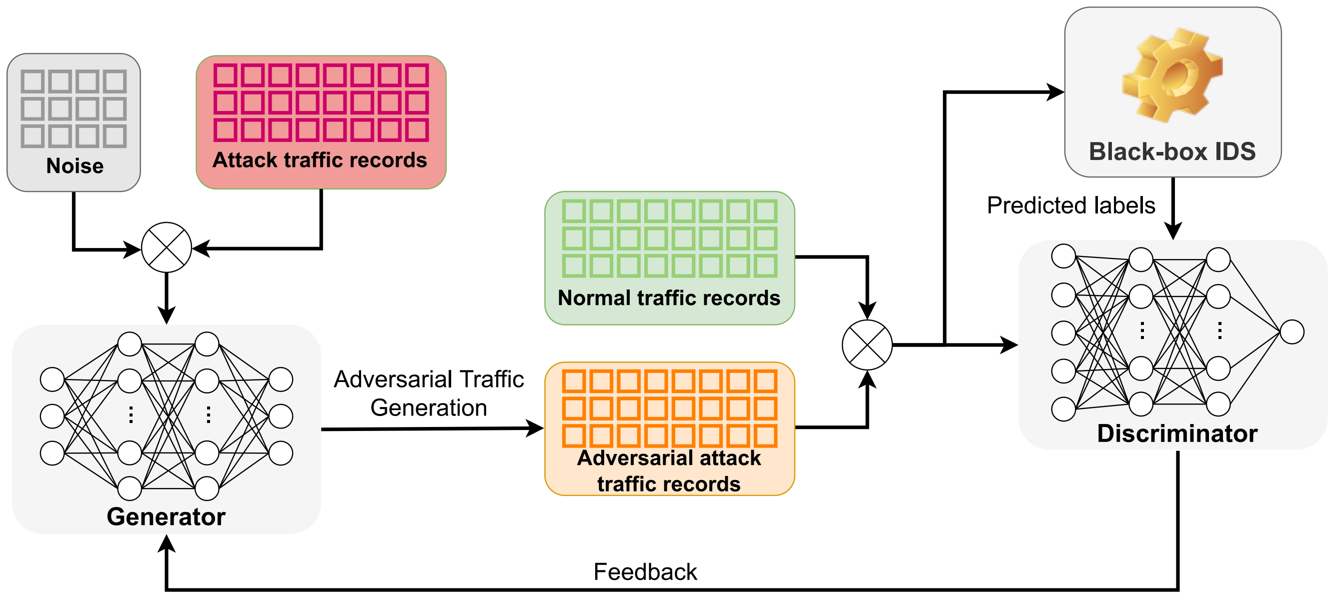


Figure 3.4: A schematic illustration of the proposed WGAN-based framework.

network attacks in reality. It is evident that each category of attack has its specific functional features representing the basic functionality of such a type of attack. It indicates that, in adversarial record generations, the attack attribute would remain unaltered if we solely fine-tuned nonfunctional features.

The generator is a crucial component of the WGAN framework as it plays the role of generating adversarial attack traffic records. A latent noise vector was concatenated with the non-functional features of the original attack traffic records before feeding the records into the generator. The noise vector contains a specific number of random noises following a $N(0, 1)$ distribution. Adding the latent noise vector aims to favour the generation of the adversarial records and meanwhile provides diversity to the generated records up to a certain level. The structure of the generator used herein is a deep neural network composed of five layers, one input layer, one output layer and three hidden layers. The ReLU activation function was utilized to add the non-linearity to the first four layers. In addition, mixed activation functions were employed at the output layer to regulate the generated records. The Sigmoid activation function was adopted to generate the normalized continuous values (range from 0 to 1), and the Gumbel Softmax function was used to produce the one-hot encoded features. The discriminator is also a five-layer deep neural network to classify attack records and benign ones. For the discriminator, the LeakyReLU function was used to enhance the

Table 3.4: Architecture of the generator and discriminator in the WGAN framework.

Operation	**Input Units**	**Output Units**	**Non-linearity**
Generator			
Linear	input_dim	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	g_output_dim	Mixed
Discriminator			
Linear	input_dim	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim//2	LeakyReLU
Linear	input_dim//2	1	

non-linearity of the neural network before output. Without knowing the structure and parameters of the black-box IDS, it is assumed that the real-time classification results of the IDS can be obtained by querying. Hence, the discriminator can gradually learn and imitate the performance of the black-box IDS. The details of the structure of the generator and discriminator are shown in Table 3.4.

For the WGAN framework, the generator and discriminator are trained and optimized alternately. The prediction outcomes of the discriminator provide the gradient information to the generator. Thus, the loss function of the generator is defined in Eq. (3.1) :

$$L_G = \mathbb{E}_{M \in S_{attack}, N} D(G(M, N)) \tag{3.1}$$

where S_{attack} is the original attack traffic; G and D represent the generator and the discriminator, respectively. The M and N denotes the non-functional features and latent noise vector, respectively. To optimize the performance of the generator (e.g., fool the black-box IDS), the training tried to minimize the loss function L_G. The discriminator optimized its performance using the predicted labels from the black-box IDS, aiming to capture the functionality of the black-box IDS. The loss function for the discriminator is given by Eq. 3.2 :

$$L_D = \mathbb{E}_{s \in B_{normal}} D(s) - \mathbb{E}_{s \in B_{attack}} D(s) \tag{3.2}$$

where s is the training set of the discriminator; B_{normal} and B_{attack} represent the predicted normal and attack traffic records, respectively, labeled based on the prediction from the black-box IDS. The outline of the general WGAN training procedure is shown in Algorithm 1. Note that for a Wasserstein GAN, in each iteration the parameters of the discriminator are updated n_d times ($n_d = 5$ in the present study) before starting to optimize the generator. This is because the generator is optimized based on the feedback of the discriminator, and it is expected to get a relatively optimal discriminator before updating the generator.

3.3 Conditional Wasserstein GAN Based Framework

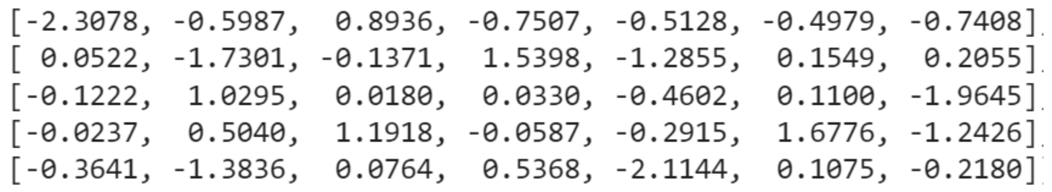
3.3.1 Input Conditions

As we discussed in the previous chapter, the class label can be used as input conditions for both the generator and discriminator. In the present study, instead of using a one-hot vector of the class labels as condition information, we adopt a word-embedding vector. The word embedding was done using the built-in function "nn.Embedding" from the PyTorch package. The advantage of using a word-embedding vector is that you can use an embedded vector to represent a specific number of words (labels) if the length of the vector is larger than the number of labels. For example, there are five labels herein representing benign traffic and four types of malicious traffic, and hence the one-hot vector will have a length of five (e.g., [1 0 0 0 0], [0 1 0 0 0], . . . , [0 0 0 0 1] for five class labels). However, we can use a word-embedding vector of any length longer than five to represent the five labels, and the value of each element in the vector is calculated using pre-defined embedding algorithms. An example of embedded vectors is shown in Figure 3.5,

Compared with one-hot encoded vectors, using the embedding vectors (e.g. embedded class labels) can make the training process of the CWGAN more smooth and stable. For instance, the generator and discriminator can update the parameters related to input conditions in a more flexible way as each element in the embedded vector is non-zero. In contrast, for one-hot encoded vectors, only the parameter

Algorithm 1 Wasserstein GAN-based framework

```
Require:
    $S_{normal}, S_{attack}$: The original records;
    $M$: The non-functional features of $S_{attack}$;
    $N$: The latent noise vector;
    $B$: The trained black-box IDS;
    $n_{epoch}$: The total number of epoch;
    $n_d$: the number of iterations of the discriminator per generator iteration.
Ensure:
    The generator $G$ and discriminator $D$ are optimized.
 1: Initialize the generator $G$ and the discriminator $D$
 2: for $i = 0$ to $n_{epoch}$ do
 3:     $M' = Generate(M, N)$
 4:     Form $S_{adversarial}$ using $S_{attack}$ and $M'$
 5:     Form a training set using $S_{adversarial}$ and $S_{normal}$
 6:     $B$ labels the training set as normal or attack
 7:     $D$ scores the training set labeled by $B$
 8:     Update parameter of $D$ according to Eq. 3.2
 9:     if $(i \bmod n_d == 0)$ then
10:         $D$ scores $S_{adversarial}$
11:         Update parameter of $G$ according to Eq. 3.1
12:     end if
13: end for
```



```
[-2.3078, -0.5987,  0.8936, -0.7507, -0.5128, -0.4979, -0.7408]
[ 0.0522, -1.7301, -0.1371,  1.5398, -1.2855,  0.1549,  0.2055]
[-0.1222,  1.0295,  0.0180,  0.0330, -0.4602,  0.1100, -1.9645]
[-0.0237,  0.5040,  1.1918, -0.0587, -0.2915,  1.6776, -1.2426]
[-0.3641, -1.3836,  0.0764,  0.5368, -2.1144,  0.1075, -0.2180]
```

Figure 3.5: An example of embedded vectors for class labels. Note that five class labels are embedded into five unique vectors.

with respect to the non-zero element of each class type will impact the output of the generator and discriminator.

3.3.2 From WGAN to CWGAN

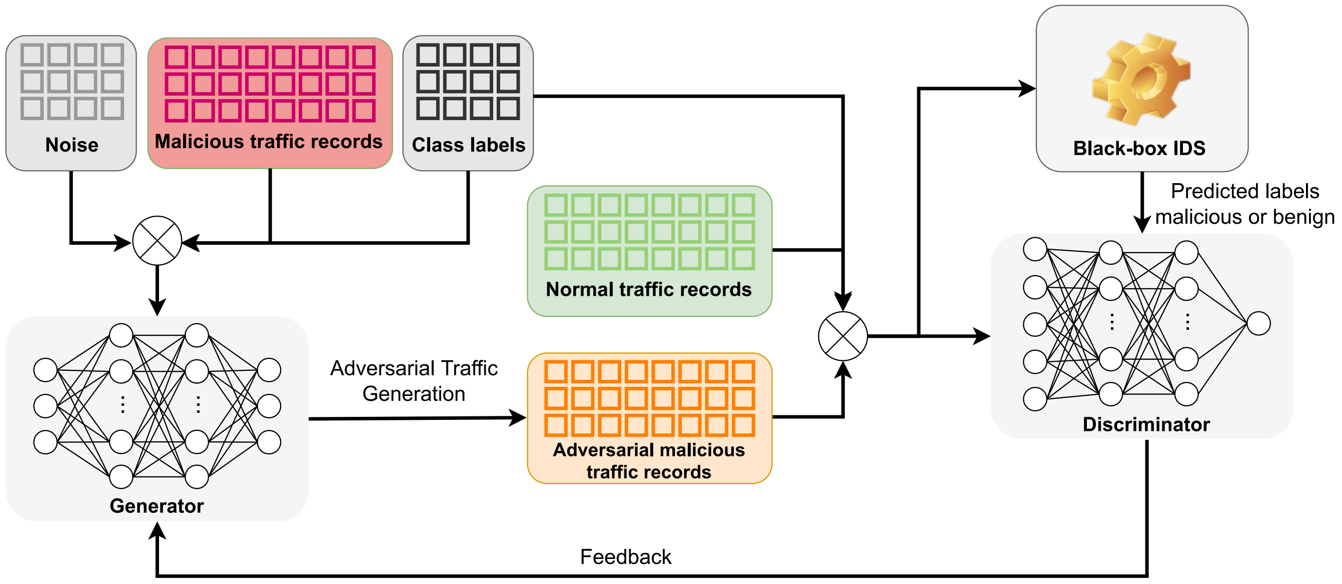

Noise
Malicious traffic records
Class labels
Black-box IDS
Predicted labels malicious or benign
Normal traffic records
Adversarial Traffic Generation
Generator
Adversarial malicious traffic records
Discriminator
Feedback

Figure 3.6: Schematic illustration of the CWGAN-based framework. Note that the embedded class labels are sent to both generator and discriminator as input conditions.

The schematic illustration of conditional Wasserstein GAN (CWGAN) is shown in 3.6. The embedded class labels are first concatenated with the noise vector and the non-functional feature vector, and then the obtained vector is fed into the generator for adversarial sample generation. Before sending the produced sample (a traffic flow) to the discriminator for scoring, the same embedding vector will be concatenated with the sample. In such a way, both the generator and discriminator will obtain the same condition information regarding a specific input record. Training with the

Table 3.5: Architecture of the generator and the discriminator in the conditional WGAN framework. Note that the number of neurons of the input layer for both generator and discriminator is increased accounting for the input conditions.

Operation	**Input Units**	**Output Units**	**Non-linearity**
Generator			
Linear	input_dim+embedding_dim	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	input_dim//2	ReLU
Linear	input_dim//2	g_output_dim	Mixed
Discriminator			
Linear	input_dim+embedding_dim	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim*2	LeakyReLU
Linear	input_dim*2	input_dim//2	LeakyReLU
Linear	input_dim//2	1	

input conditions (embedded class labels), the GAN can learn which specific type of data it is dealing with for both generative and discriminative tasks, and hence after optimization, the generator obtains the capability of generating the specific type of samples as required (controlled by using embedded class labels).

The update algorithm for CWGAN is shown in Algorithms 2. Note that in lines 3, 7 and 10, the input data is sent into the generator and discriminator along with the input condition E.

The modified architectures of the generator and the discriminator are shown in Table 3.5. Note that here we only increase the number of units for the input layer of the generator and the discriminator accordingly, but keep the number of units of the hidden layers unchanged. This is because we will compare the performance of the CWGAN with WGAN, and increasing the size of all hidden units will significantly alternate the capability of the generator and discriminator.

Algorithm 2 Conditional Wasserstein GAN-based framework

```
Require:
    $S_{normal}, S_{attack}$: The original records;
    $M$: The non-functional features of $S_{attack}$;
    $N$: The latent noise vector;
    $E$: The embedded class labels;
    $B$: The trained black-box IDS;
    $n_{epoch}$: The total number of epoch;
    $n_d$: the number of iterations of the discriminator per generator iteration.
Ensure:
    The generator $G$ and discriminator $D$ are optimized.
1:  Initialize the generator $G$ and the discriminator $D$
2:  for $i = 0$ to $n_{epoch}$ do
3:      $M' = Generate(M, N, E)$
4:      Form $S_{adversarial}$ using $S_{attack}$ and $M'$
5:      Form a training set using $S_{adversarial}$ and $S_{normal}$
6:      $B$ labels the training set as normal or attack
7:      Distance score = Discriminator(Labeled training set, $E$)
8:      Update parameter of $D$ according to Eq. 3.2
9:      if ($i$ mod $n_d == 0$) then
10:         Distance score = Discriminator($S_{adversarial}$, $E$)
11:         Update parameter of $G$ according to Eq. 3.1
12:     end if
13: end for
```

Chapter 4

Performance Analysis

In this chapter, we first introduce the CICIDS2017 dataset, which was used to test the performance of the designed GAN and CGAN-based attack methodology. Then, we review the selected eight ML and DL algorithms. Before showing our results, we depict the experiment setup and the evaluation metric used for the IDSs and adversarial attack framework. Lastly, we present our testing results for both GAN and CGAN method and also give a thorough discussion.

4.1 CICIDS2017 Dataset

4.1.1 Dataset Selection

There are a number of different datasets that have been published for intrusion detection investigations. The first well-known dataset used to evaluate ML/DL-based IDSs is KDD-Cup 99 dataset [48]. However, KDD-Cup 99 dataset has a huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevents them from learning unfrequent records, which could be more harmful to networks [31] [44]. Also, it has been reported that using KDD-Cup 99 dataset leads to the overestimation of the performance of some anomaly detection techniques [38]. To overcome the drawbacks of the KDD-Cup 99 dataset, Tavallaee et al. did a statistical analysis of the KDD-Cup 99 dataset. They deleted repeated data entries and select records from the KDD-Cup 99 dataset based on "level of difficulty" to form a new dataset, named NSL-KDD. The NSL-KDD dataset does solve some issues of the KDD Cup 99 dataset (i.e., no duplicate and redundant records and a more reasonable size of the train and test dataset). However, as the NSL-KDD is an optimized subset of KDD-Cup 99, it still has some inherent problems inherited from KDD-Cup 99. For instance, McHugh [39] criticizes that the NSL-KDD does not represent actual existing networks and associated attack scenarios.

CICIDS2017 dataset contains benign records and the most up-to-date common attack records, which resembles the true real-world data [49]. In addition, CICIDS2017 includes the results of the network traffic analysis performed by CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack. Unless the NSL-KDD dataset, CICIDS2017 contains a sufficient number of records of unfrequent low-footprint attacks. Though CICIDS2017 has some minor drawbacks such as containing a few NULL values, it is has become more and more popular when studying IDSs.

4.1.2 Dataset Description

The CICIDS2017 Dataset [49] is adopted herein to test and evaluate our proposed WGAN-based attack methodology. The dataset was generated using two pre-built network infrastructures as shown in Figure 4.1. Therein, one network is treated as the victim corporation subjected to network attacks (30 servers with hundreds of client machines), while the other network plays the role of the attacker (50 machines).

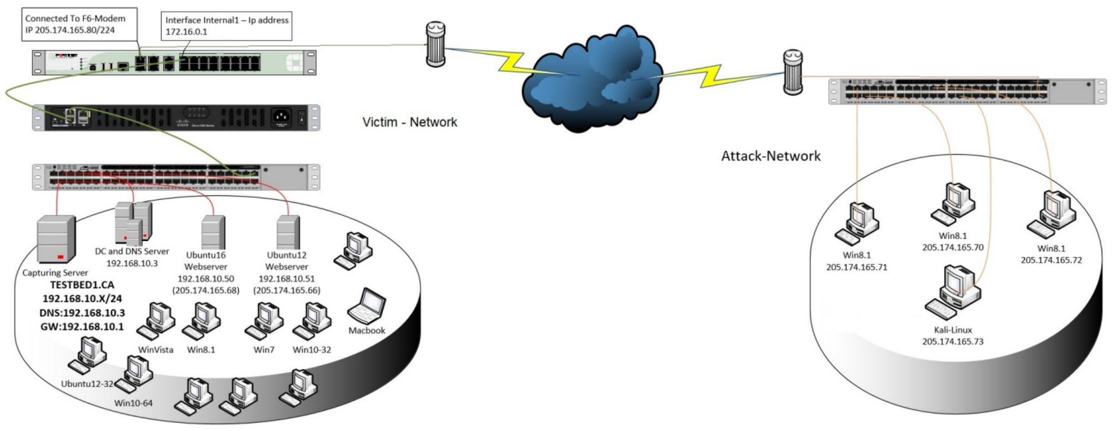


Figure 4.1: An illustration of the network infrastructures used to generate the CICIDS2017 dataset [49].

The dataset composes different scenarios of fourteen modern attack types along with normal network traffic. Each entry of the dataset consists of 80 columns, which are flow features extracted from network traffic. The number of entries for normal and different types of attack records in the dataset is shown in Table 4.1. In the present study, we only concentrate on four major attack types from the dataset, namely, DoS Hulk, PortScan, DDoS, and DoS GoldenEye, which compose more than 95% of the total malicious records.

Table 4.1: Type of attack traffic records in CICIDS2017.

Type of Attack Traffic	**No. of Data**
Benign	2273097
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS Slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack: Brute Force	1507
Web Attack: XSS	652
Web Attack: Brute Force	36
Web Attack: Sql Injection	21
Heartbleed	11

4.1.3 Data Preprocessing

The data types of the features in CICIDS2017 can be either continuous or discrete. A preprocessing was performed for the dataset to transform the data into valid inputs for the WGAN-based framework. The data preprocessing includes several sub-stepds. Firstly, several irrelevant columns that have a constant value were dropped, as they do not play any roles in differentiating categories. Then, the rows that contain null or infinitive values were deleted from the dataset. They only compose a small part (hundreds of entries) of the entire dataset, and hence, will not impact the results. Afterwards, the feature values were all normalized to a range between 0 and 1 (i.e., min-max normalization) using Eq. 4.1, given as:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Performing normalization is because the value range of different features is widely varied. For example, the "Total Fwd packets" feature ranges from 1 to 219759, while the "Down/Ratio Ratio" feature only ranges from 0 to 156. The normalized data can lead to better training and test results. Lastly, there are several binary features related to flag count, and they are one-hot encoded.

4.1.4 Functional Features

For the CICIDS2017 dataset, Sharafaldin et al. [49] performed an analysis using a regressor and selected four characteristic features for each attack category. The characteristic features of the selected attack categories for the present study from the CICIDS2017 dataset are shown in Table 4.2. In this study, all the 10 characteristic features shown in Table 4.2 are treated as functional features for each selected attack category, which means they are all kept unchanged during the adversarial traffic generation. Our decision is understandable and reasonable. First, it is always expected to modify as few features as possible when generating adversarial attack traffic to maintain the original traffic pattern and preserve the attack functionalities of the unmodified attack traffic. Second, the selected features are meaningful regarding the nature of the attack traffic. For example, DoS and DDoS attacks usually flood a system using a large amount of data within a very short period. This functional nature can be reflected by a short flow inter-arrival time (IAT) and a huge average packet size, both of which are treated as functional features herein.

4.2 Experimental Setup and Evaluation Metrics

An open-source machine learning framework, PyTorch, was used to build up the proposed architecture. Eight common ML/DL-based IDS were considered herein to evaluate the capability of the proposed framework against IDS detection. As discussed in previous chapters, the considered ML/DL-based IDSs include Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), Multilayer Perceptrons (MLP), Recurrent Neural Network (RNN), and Covolutional Neural Network (CNN). Note that the five ML algorithms (NB, LR, DT, RF, and GB) were constructed using Scikit-learn and made callable by PyTorch through a wrapper class, whereas the MLP, RNN, and CNN are built up using

Table 4.2: Characteristic features and corresponding attack types for CICIDS2017.

Characteristic Features	**Corresponding Attack Types**
Bwd packet Length Std	DoS GoldenEye, DoS Hulk, DDoS
Flow Duration	DoS Hulk, DDoS
Flow IAT Mean	DoS GoldenEye
Flow IAT Std	DoS Hulk, DDoS
Flow IAT Min	DoS GoldenEye
Fwd IAT Min	DoS GoldenEye
Average Packet Size	DDoS
PSH Flag Count	PortScan
Bwd Packets/s	PortScan
Win F.Bytes	PortScan

the PyTorch deep learning library. As mentioned previously, we concentrate on the four major types of attack traffic in the CICIDS2017, which are DoS Hulk, PortScan, DDoS, and DoS GoldenEye.

The eight IDSs were pre-trained separately as binary classifiers (benign vs. attack) before training the WGAN framework. Given that the characteristics of each attack category are different, the WGAN framework was trained to generate the adversarial samples for solely one attack category each time to deceive one selected IDS. For each attack category, half of the original data was used to train, while the rest half of the data was retained for testing. The WGAN framework is trained with a batch size of 64 for 30 epochs. The learning rates of the generator and discriminator are both 0.0001 with an RMSProp optimizer. The dimension of the latent noise vector is eight. The weight clipping threshold for the discriminator is ± 0.01. As the conditional WGAN (CWGAN) also takes the embedded class labels as input, the data of all types of attack records can be fed into the CWGAN simultaneously. Thus, we only need to train CWGAN with each ML/DL algorithm once for all types of attack records.

For IDSs, we calculate all typical evaluation metrics for a binary classifier, including accuracy (detection rate), precision, recall, F-score, the area under the receiver operating characteristic (AUROC), and average precision (AUPRC). We also present the confusion matrix of the result of each algorithm.

Regarding the WGAN and CWGAN framework, for the evaluation metrics, the detection rates of each IDS for the original and adversarial datasets are calculated. The detection rate is the same as the accuracy for a classifier, which defines as the proportion of correctly detected attack traffic records by the IDS to all of those attack records, reflecting the evasion ability of the adversarial samples and the robustness of the IDS. A lower detection rate indicates more attack traffic records evade the detection of the IDS, revealing higher adversarial attack effectiveness. Therefore, our goal is to optimize the WGAN framework to obtain a lower detection rate for each IDS.In addition to DR, the evasion increase rate (EIR) is widely adopted to evaluate the efficiency of adversarial attacks, given by:

$$EIR = 1 - \frac{Adversarial\ DR}{Original\ DR} \tag{4.2}$$

A lower EIR represents that more adversarial samples generated by the WGAN are detected by IDSs, indicating a worse performance of the WGAN framework. Therefore, we expect higher EIR can be achieved with the proposed framework.

4.3 Experimental Results

4.3.1 Performance of Selected IDS Schemes

The training of ML algorithms is relatively faster and easier compared to DL algorithms. For the sake of being concise, we herein only show the training procedure of the DL algorithms. The evolution of the batch-averaged loss of the test dataset during training for the MLP, RNN, and CNN are shown in Fig. 4.2. It is noticed that the MLP converged very rapidly during training. After two to three epochs, the batch-averaged loss maintained at a level around 12. The convergence speed of the RNN was slower than the one of the MLP, and it took about eight epochs to converge. The CNN took the longest time to converge, and after about sixteen epochs the batch-averaged loss stayed at a level around 13.5. The results are understandable as the neural networks with more complex architectures are more difficult to optimize.

The evolution of the accuracy (detection rate) of the test dataset for the three DL algorithms is shown in Fig. 4.3. After two to three epochs, the accuracy of the MLP remained at a level of 97%. Similar to the batch-averaged loss, the evolution of the detection rate of the RNN and CNN is slower. After about ten epochs, the accuracy of both algorithms stopped increasing, indicating the convergence of the RNN and

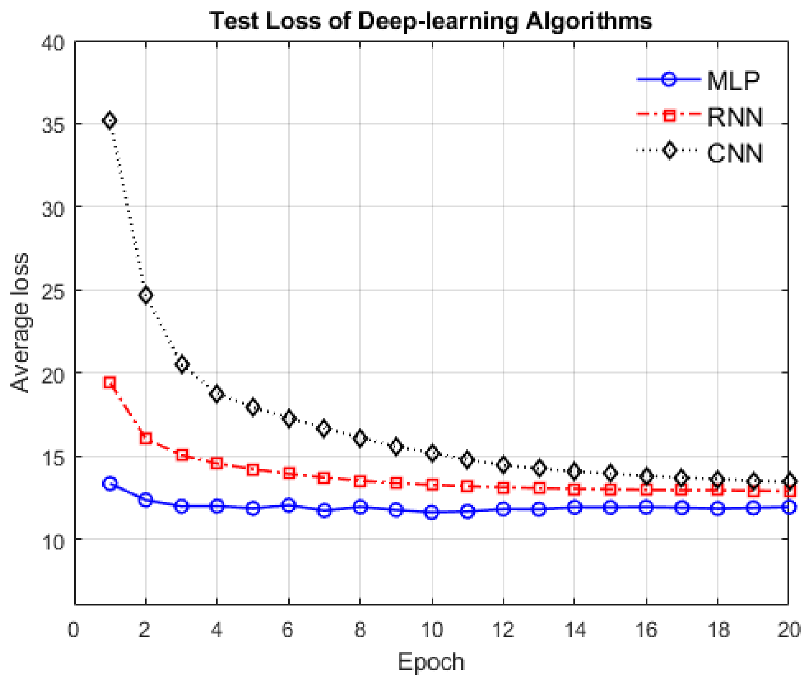


Figure 4.2: Evolution of average loss versus epoch for MLP, RNN, and CNN.

CNN. Both of the two algorithms had a detection rate of 96.7% after optimization, which is slightly slower than the MLP. As the major purpose of the present study is not to develop a perfect IDS using specific DL algorithms, we stop further optimizing the MLP, RNN, and CNN.

The confusion matrix of all selected algorithms is shown in Table 4.4. The confusion matrix reflects that it is a binary classification with a highly imbalanced dataset. There are about 600 thousand benign records and 80 thousand malicious records. The performance of the NB algorithm was very poor, and it misclassified about 140 thousand benign cases and 37 thousand malicious cases. The prediction of the LR algorithm is much better. However, the LR algorithm had a very high false negative rate, misclassifying 43 thousand of malicious cases. The total number of misclassifications for DT, RF, and GB algorithms are all less than 1000, which outperformed other algorithms. The performance of the MLP, RNN, and CNN are close, which all misclassified about 20 to 25 thousand records.

The full evaluation results for the eight selected IDSs with the test dataset is shown in Table 4.3. The NB algorithm has the worst performance, which is consistent with the results of the confusion matrix. The poor performance of the NB algorithm is

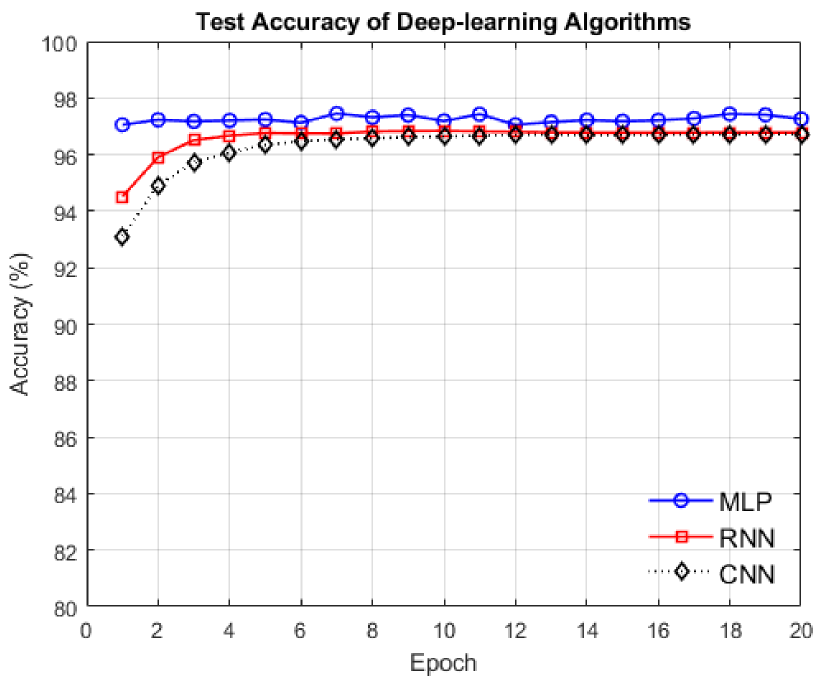


Figure 4.3: Evolution of test accuracy versus epoch for MLP, RNN, and CNN.

probably due to its simplicity (i.e., not able to capture high-dimensional characteristic features and non-linearity of the dataset). The NB and LR algorithms have the lowest precision and recall rates, respectively. The low recall rate but the high precision rate of the LR algorithm indicates that it is not good at handling imbalanced dataset. It is shown that the DT, RF, and GB algorithms have a perfect performance, and all the evaluation metrics reach nearly the highest score. For the three deep learning algorithms, the MLP has the highest accuracy and recall rates, while the CNN has the highest precision rate. The performance of the RNN lies between the MLP and the CNN. The F-score, AUROC, and AUPRC show consistent results, reflecting the overall capability of the algorithms. From the perspective of intrusion detection, RF is the clear winner among all IDSs.

4.3.2 Performance of Wasserstein GAN Based Framework

Fig. 4.5 shows the detection rate of the IDSs for the DoS Hulk attack with the original dataset and the adversarial dataset, respectively. For the original dataset, the detection rate of the NB algorithm is around 80%, while the other seven algorithms all have a detection rate higher than 97%. However, with the adversarial dataset, the

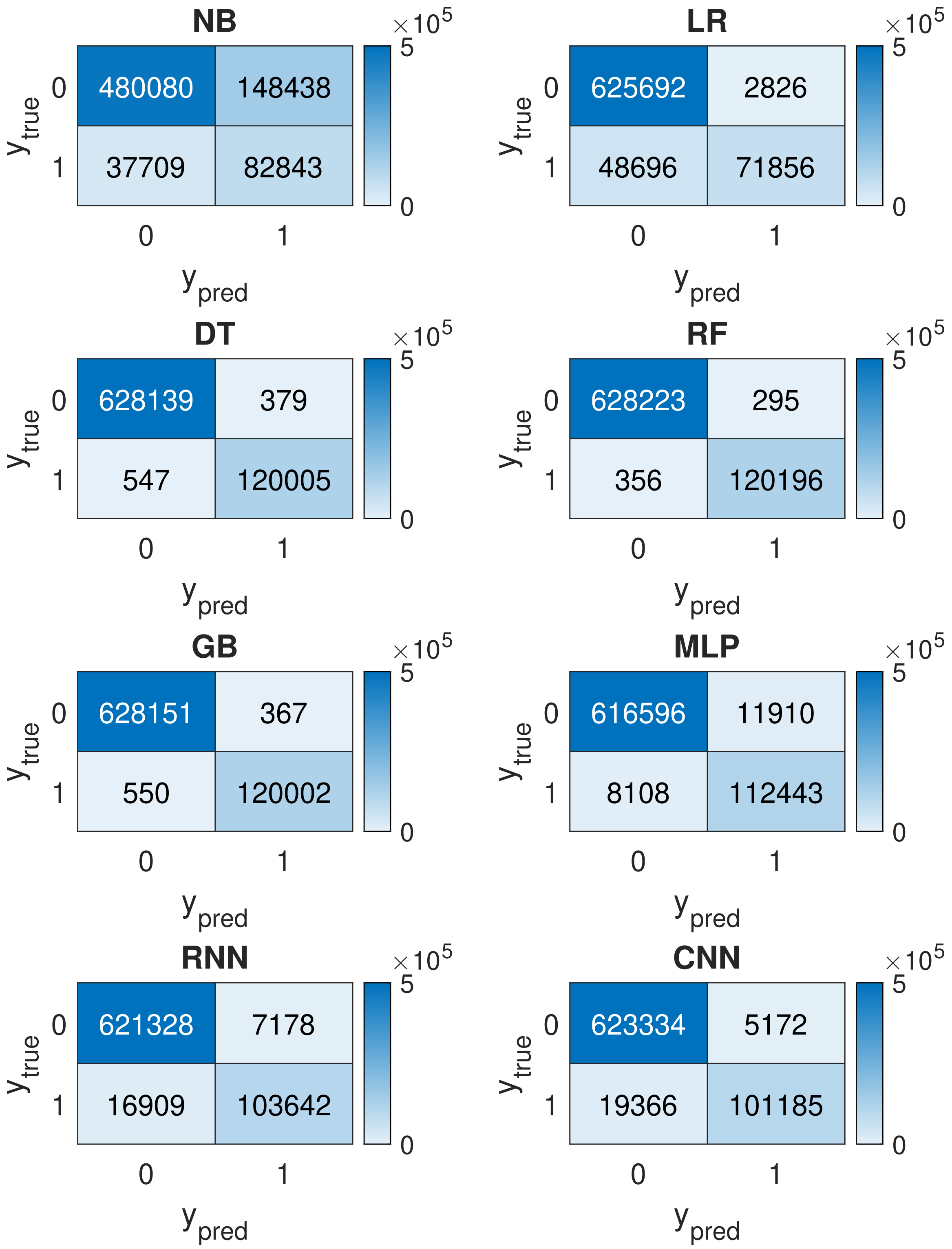


Figure 4.4: Confusion matrix for the selected ML/DL algorithms.

Table 4.3: Experimental results for selected ML/DL algorithms with CICIDS2017 dataset.

IDS	Accuracy	Precision	Recall	F-score	AUROC	AUPRC
NB	75.15%	35.82%	68.72%	0.471	0.842	0.538
LR	93.12%	96.22%	59.61%	0.736	0.974	0.897
DT	99.88%	99.68%	99.55%	0.996	0.998	0.994
RF	99.91%	99.75%	99.71%	0.997	1.000	1.000
GB	99.88%	99.69%	99.55%	0.996	0.999	0.995
MLP	97.33%	90.42%	93.27%	0.918	0.997	0.983
RNN	96.80%	93.41%	86.22%	0.897	0.995	0.978
CNN	96.72%	95.14%	83.94%	0.892	0.993	0.973

detection rate of the NB, LR, RF, MLP, and RNN algorithms all decreased to zero. The GB algorithm can slightly resist adversarial attacks, having a detection rate of 22% for the generated dataset. The CNN and DT have a strong capability to detect adversarial records, with a detection rate of 65% and 98%, respectively.

The detection rate of the IDSs with the original and adversarial datasets for the DDoS attack is shown in Fig. 4.6. The performance of the selected IDSs for the original DDoS samples is similar to the one for the unmodified DDoS Hulk samples. However, the robustness of the IDS under DDoS adversarial attack is distinct. Figure 4.6 shows that the MLP and CNN was able to detect more than 90% of the generated DDoS samples. The DT and GB algorithms also have some ability to distinguish the adversarial samples from the real records, having a detection rate up to 18% and 16%, respectively.

It is shown in Fig. 4.7 that all the IDSs performed very well on the detection of the original data entries of the DoS GoldenEye records except the NB algorithm, which only has a detection rate less than 78%. The DT and GB algorithms perfectly detected the adversarial samples of the DoS GoldenEye attack. The MLP and CNN also show compromised results when defending against adversarial attacks, having an adversarial detection rate about 83% and 78%, respectively.

The performance of the IDSs for the original dataset of the PortScan records is distinct compared to one of the IDSs for other types of attack records. The NB and

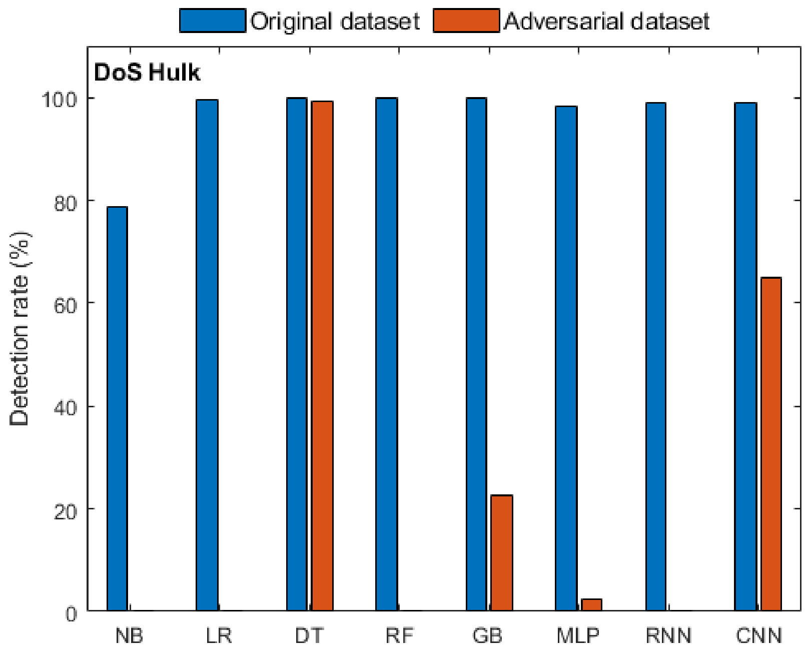


Figure 4.5: The detection rate of the selected ML/DL algorithms with the original and adversarial datasets for DoS Hulk.

LR algorithms can barely detect the PortScan records from the original dataset, while the other IDSs have very high detection rates (e.g., detection rate higher than 97%). Regarding the adversarial data set, only the CNN has a good ability to defend against the adversarial attack with a detection rate around 60%. The DT and GB algorithms captured a few adversarial samples, having a detection rate around 8% on generated records.

The EIR(%) can more directly reflect the relative power between the IDSs and the WGAN framework, which is shown in Table 4.4. The NB, LR, RF, and RNN algorithms are completely fooled by the proposed framework (i.e., with a 100% EIR). The DT algorithm can perfectly detect the adversarial samples of the DoS Hulk and DoS GoldenEye attacks, while the MLP is good at defending the adversarial attacks from the DDoS and DoS GoldenEye samples (i.e., with a 7.2% and 14.53% EIR, respectively).

With the adversarial dataset, the performance of the IDSs is significantly impacted. Many IDSs can be fully deceived by the adversarial attack samples (i.e., note the zero detection rates and 100% EIR values in Table 4.4). However, there are several

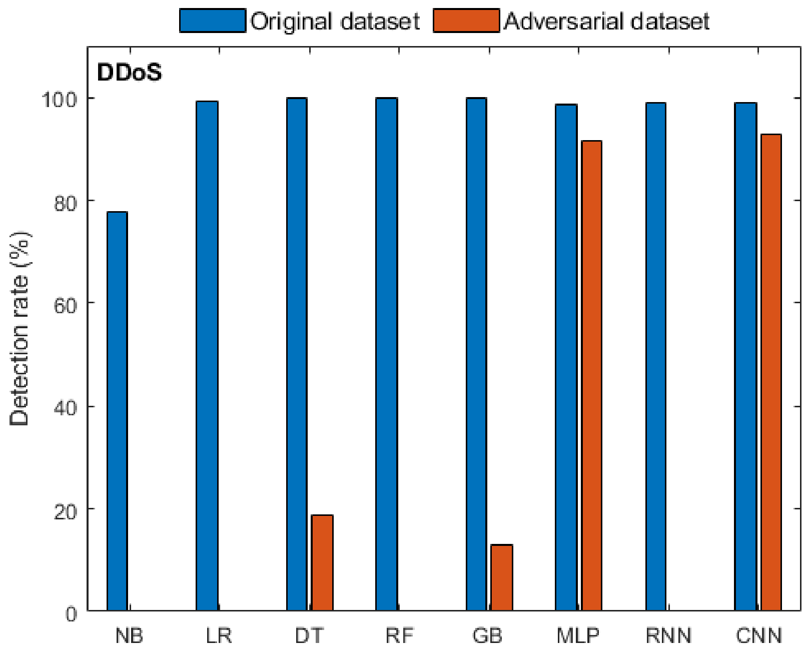


Figure 4.6: The detection rate of the selected ML/DL algorithms with the original and adversarial dataset for DDoS.

exceptions, which can be summarized as follows: The DT algorithm shows the great capability to detect the adversarial samples of DoS Hulk and DoS Goldeneye records (see the EIR in 4.4); for the adversarial samples of DoS GoldeEye, the detection rate of the GB and MLP are also promising, with a detection rate about 98% and 85%, respectively (see Fig. 4.7); also, the MLP is very robust under the adversarial attacks of the DDoS records, having a detection rate higher than 90 % (see Fig. 4.6); lastly, regarding the GB algorithm, it shows slight resistance with the adversarial samples of DoS Hulk and DDoS attack records (see Table 4.4).

Based on authors' knowledge, there has been no approach reported in the literature that can explain the distinct performance outcomes of IDSs under adversarial samples. However, it must highly related to the natural of the ML algorithms. For example, the DT algorithm only classify the records using selected features based on information gains, and the modified non-functional features may not be selected by the algorithms, and hence will not cause impact on the prediction results. With respect to the GB algorithms, the boosting ensemble method seems provide the robustness to the algorithm against adversarial samples. The explainability of the WGAN framework

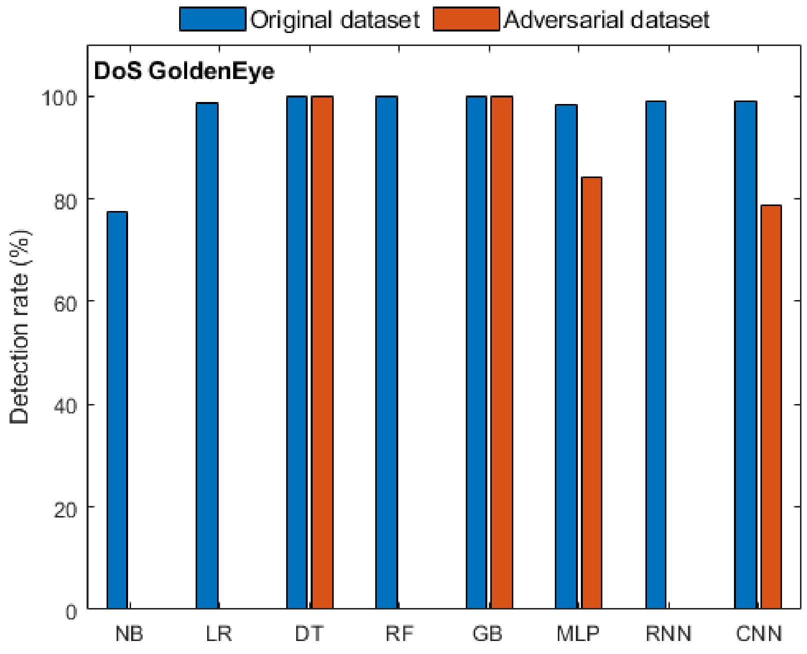


Figure 4.7: The detection rate of the selected ML/DL algorithms with the original and adversarial dataset for DoS GoldenEye.

will be left to the future work.

Table 4.4: The EIR(%) of the IDSs for adversarial samples of DoS Hulk, DDoS, DoS GoldenEye, and PortScan generated by WGAN.

Attack Type	**NB**	**LR**	**DT**	**RF**	**GB**	**MLP**	**RNN**	**CNN**
DoS Hulk	100	100	0.8	100	77.48	97.46	100	34.34
DDoS	100	100	81.28	100	86.89	7.2	100	6.06
DoS GoldenEye	100	100	0.1	100	0.1	14.53	100	20.53
PortScan	100	100	94.49	100	95.1	100	100	38.05

4.3.3 Performance of Conditional Wasserstein GAN Based Framework

The results of the CWGAN method are present in this section and are also compared with results from the original framework based on WGAN.

Figure 4.9 shows the detection rate of the Naive Bayes algorithm for the original data sets and the adversarial data set from the WGAN and CWGAN for all four types

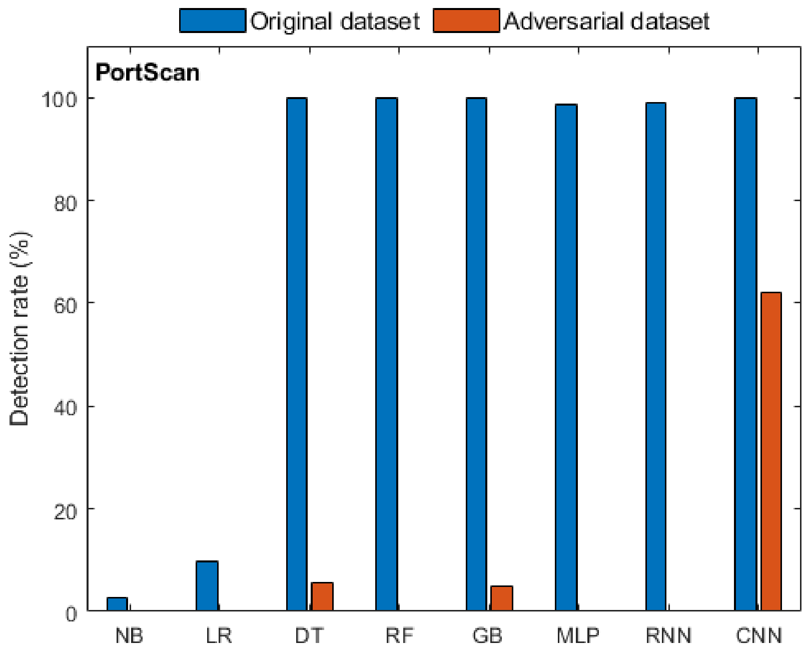


Figure 4.8: The detection rate of the selected ML/DL algorithms with the original and adversarial datasets for PortScan.

of attacks. The NB algorithm barely detected any adversarial records generated by either the CGAN framework or the WGAN framework. This is understandable as the NB algorithm is simple and assumes that the value of a traffic feature is independent of the value of any other features, indicating that the NB algorithm can not capture the relationship among flow features and is easy to be deceived. Also, the NB algorithm has no hyperparameters, and hence, there is no way to play around to further improve the performance of the NB algorithm. Though the NB algorithm is efficient and has been used in some real-world classification tasks,it should not be used in network intrusion detection.

Figure 4.10 shows that though the Logistic Regression algorithm have better performance than the NB algorithm on detection of the original attack records, it did not provide any improvement for the IDS in defending against the adversarial attacks (i.e., the detection rate for adversarial samples was close to zero for all types of attack records). The only hyper-parameter of the LR algorithms is its regularization term C, which can prevent the overfitting of the algorithm on training. But an extra regularization term is not able to provide any extra capability to the algorithm itself.

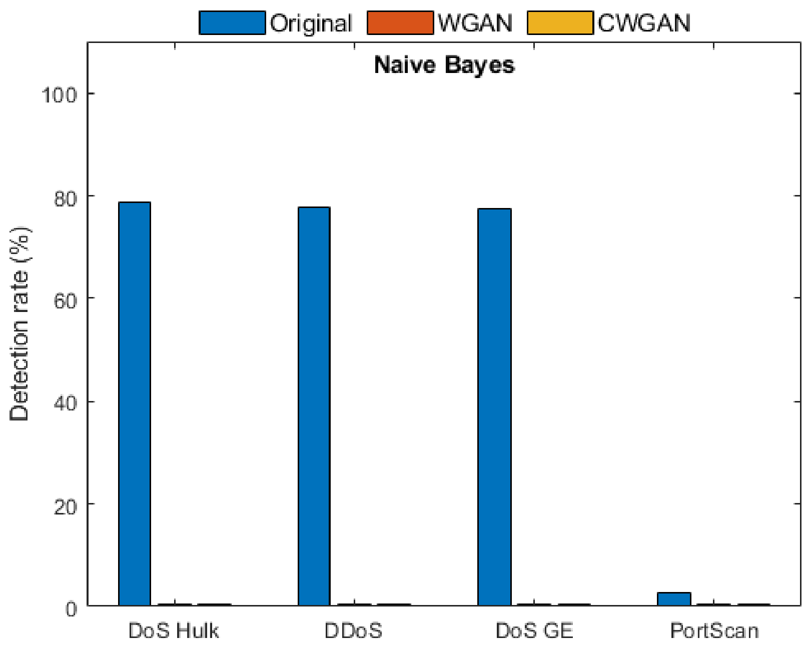


Figure 4.9: The detection rate of the Naive Bayes algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that the NB algorithm is not able to detect any generated adversarial records from both the WGAN and CWGAN frameworks.

In other words, the LR is a generalized linear algorithm, and hence, the GAN could learn the impact of each feature on the algorithm output and manipulate the features to affect the algorithm's prediction accordingly. Results in Fig. 4.10 indicate that though the LR algorithm is good to be used as a benchmark in some cases when constructing prototypes of IDSs(i.e., the LR algorithm has a detection rate higher than 98% for DoS Hulk, DDoS and DoS GoldenEye attacks), it should not be used in the final IDSs due to its poor performance under adversarial attacks.

Figure 4.11 shows the detection rate of the Decision Tree algorithm on the original records and adversarial records for the four attack types. For the DoS GoldenEye attack, the DT algorithm can detect the generated records from both the WGAN and the CWGAN. The DT algorithm has a slightly higher detection rate on the attack records generated by the WGAN than the ones produced by the CWGAN for the DDoS and the PortScan attack. However, when it comes to DoS Hulk, the DT algorithm can detect the adversarial records crafted by the WGAN but cannot defend against the adversarial attack by the CWGAN. Therefore, for the CICIDS2017

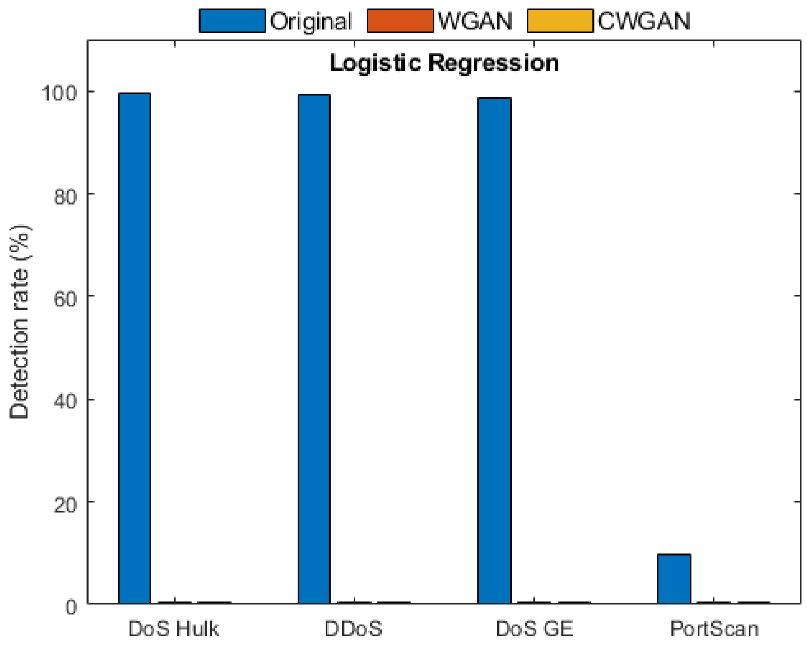


Figure 4.10: The detection rate of the Logistic Regression algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that though the LR algorithm is good at detecting DoS Hulk, DDoS, and DoS GE attacks, it failed to detect adversarial samples of the three attack types.

dataset and the DT algorithm in the present study, the CWGAN framework has stronger attack capbility. Intuitively, as the CWGAN is handling a more challenging task (e.g., generating multiple types of adversarial attacks simultaneously), the adversarial records generated by the CWGAN framework should be easier to be detected. One possible reason is that to account for the input condition information (e.g. embedded class labels) the number of neurons in the input layer is increased for both the generator and the discriminator, and hence, the capability of the sub-components is enhanced. Also, by providing the class information, the CWGAN frame might be able to learn how to craft adversarial samples based on the type of attack records.

The results of the IDS using the Random Forest algorithm are shown in Fig. 4.12. It is noticed that the RF algorithm has a perfect performance in detecting the original attack records but failed to detect the adversarial records crafted by the WGAN and the CWGAN framework. Thus, how robust an IDS is under adversarial attacks might have no relationship with its capability to detect original unmodified attack records. Also, it is well known that the RF algorithm is an ensemble learner constructed based

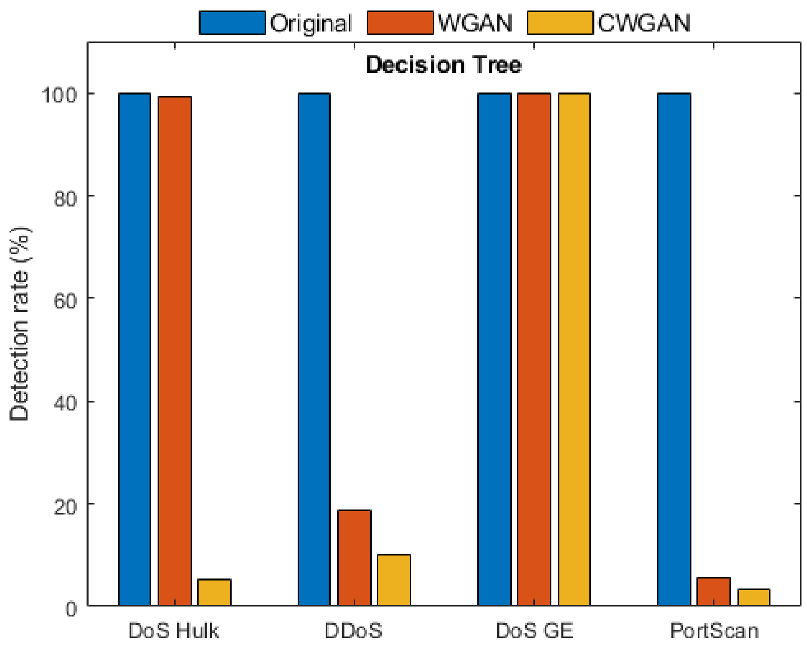


Figure 4.11: The detection rate of the Decision Tree algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that the distinct performance for the DT algorithm on detecting the two different adversarial dataset generated by the WGAN and the CWGAN algorithm.

on a bagging strategy using a number of DT algorithms. In general, the RF algorithm is more complex than the DT algorithm. However, the more sophisticated RF algorithm did not show expected robustness in defending against adversarial attacks. Therefore, we can not make a conclusion that an IDS with more complex algorithms will have a better performance when subjecting to adversarial attacks. It is worth mentioning that in the present study one hundred weaker classifiers (e.g., Decision Tree) were used to build up the RF algorithm. If the number of weaker classifiers will affect the robustness of RF against adversarial attacks will be left to further study.

Figure 4.13 shows the detection rate of the Gradient Boosting algorithm on the original and the adversarial records of all types of attacks considered. Regarding the adversarial attacks, the effect of the WGAN and the CWGAN is consistent with the IDS based on the GB algorithm. By comparing the results in Figs. 4.12 and 4.13, it reflects that the boosting ensemble strategy outperforms the bagging ensemble strategy in defending against adversarial attacks (i.e., higher detection rate on adversarial

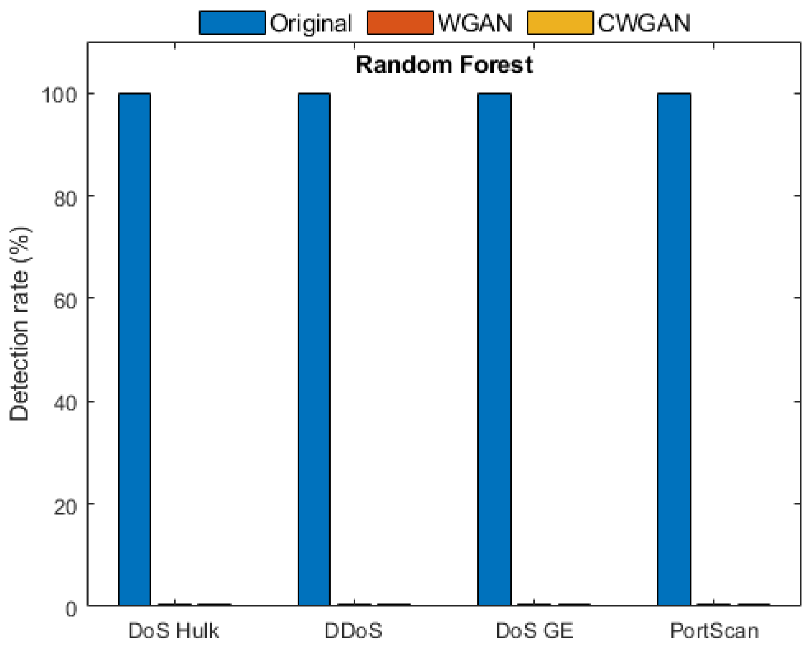


Figure 4.12: The detection rate of the Random Forest algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that though the RF algorithm provides nearly perfect results on detection the original dataset, it can barely detect any generated data samples.

samples of different attacks). Note that the GB algorithm has a number of hyperparameters such as learning rate, number of estimators, and minimum sample splits. By tuning up these hyperparameters, the capability of the GB algorithm on capturing the adversarial samples could be further enhanced. Also, Gradient Boosting is the most common sort of widely used boosting algorithm, and other more complicated boosting algorithms such as XGBoost [14] and Light GBM [26] will be tested and reported in a future publication.

As the most fundamental and widely used deep learning algorithm, the MLP has been employed to build up the IDS. The detection rate of the MLP algorithm on the original dataset and two adversarial datasets is shown in Fig. 4.14. The MLP shows its robustness against adversarial samples of the DDoS and the DoS GE attacks from both the WGAN and the CWGAN frameworks. However, the MLP algorithm did not perform well in detecting adversarial samples of the other two types of attack (e.g., DoS Hulk and PortScan), and especially for the modified PortScan samples, the detection rate of the MLP algorithms is close to zero. In the present study, both

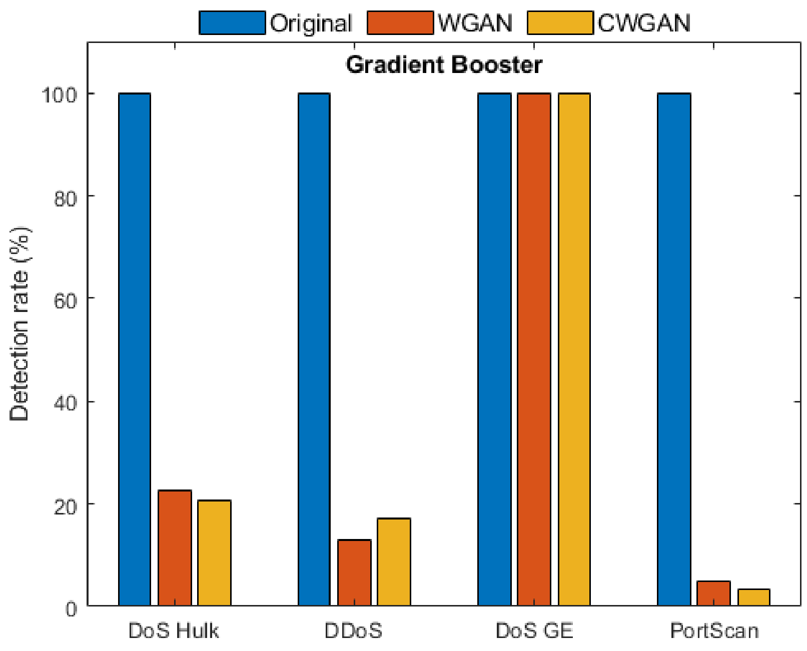


Figure 4.13: The detection rate of the Gradient Boosting algorithm with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that the performance of the GB algorithm is generally consistent on the adversarial datasets produced by the WGAN and the CWGAN.

the generator and the discriminator were constructed as a deep neural network, and thus, the performance of the IDS based on the MLP algorithm can be treated as a benchmark for other machine learning algorithms. It is known that the capability of the MLP can be adjusted by increasing the number of hidden layers and the number of neurons in each layer. Alternating the architectures of either the MLP or the generator and discriminator could impact the final results.

The recurrent neural network is another very popular deep learning algorithm and its test performance is shown in Fig. 4.15. Though the detection rate of the RNN on the original dataset is very high (97.5%), the RNN is not able to defend the adversarial attacks under both the WGAN and CWGAN framework. In contrast to the MLP, the RNN did not show any resistance to adversarial attacks. This is probably because the input data (either modified or unmodified) is not a real time sequence (i.e., considered as a pseudo time sequence with only one time step), and the RNN can not maximize its capability. Again, the performance of the RNN on the adversarial samples indicates that the complicity of the algorithms is not related

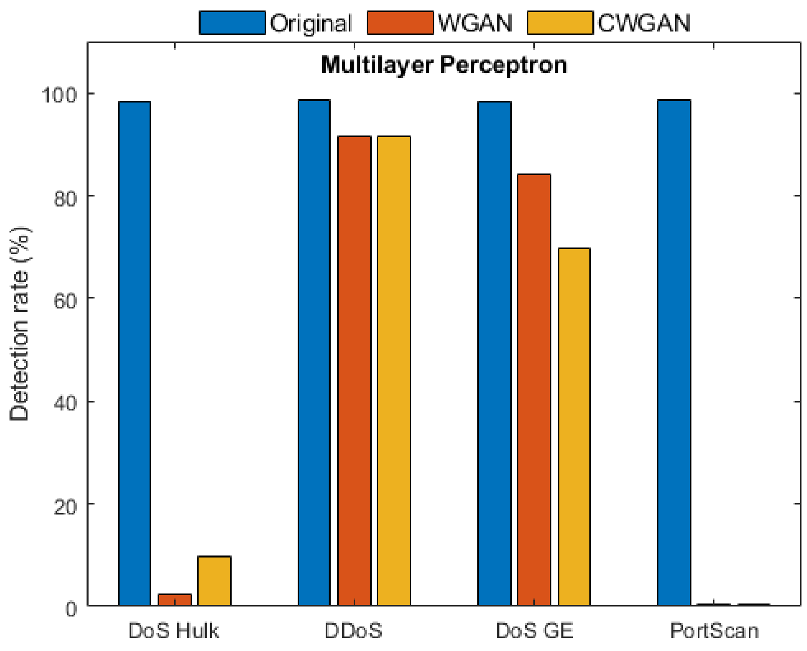


Figure 4.14: The detection rate of the Multilayer Perceptron with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that the MLP performed very well on detecting the adversarial samples of the DDoS and DoS GoldenEye attacks.

to the algorithm's robustness when subjecting an adversarial attack.

The results of the CNN in detecting the original and adversarial datasets of four types of attack records is shown in Fig. 4.16. Though the CNN is not the best one in detecting the original traffic records, it is generally the most robust one when defending against adversarial attacks from either WGAN or CWGAN. The detection rate of the CNN on adversarial samples is overall above 60%, except for the DoS Hulk samples from the CWGAN (¿ 50%). Furthermore, the performance of the CNN in detecting the adversarial datasets is consistent between the WGAN and CWGAN framework, reflecting its resilience and stability. The robustness of the CNN is possibly due to two reasons. On the one hand, using a regular deep neural network as a discriminator may not be able to mimic the feature extracting characteristics (e.g., the convolution and max pooling procedure) of the CNN, and hence, the generator cannot get efficient feedback from the discriminator to improve its capability on sample generation. On the other hand, as the CNN algorithm has the ability to extract features from the records, a generator might be difficult to deceive the CNN

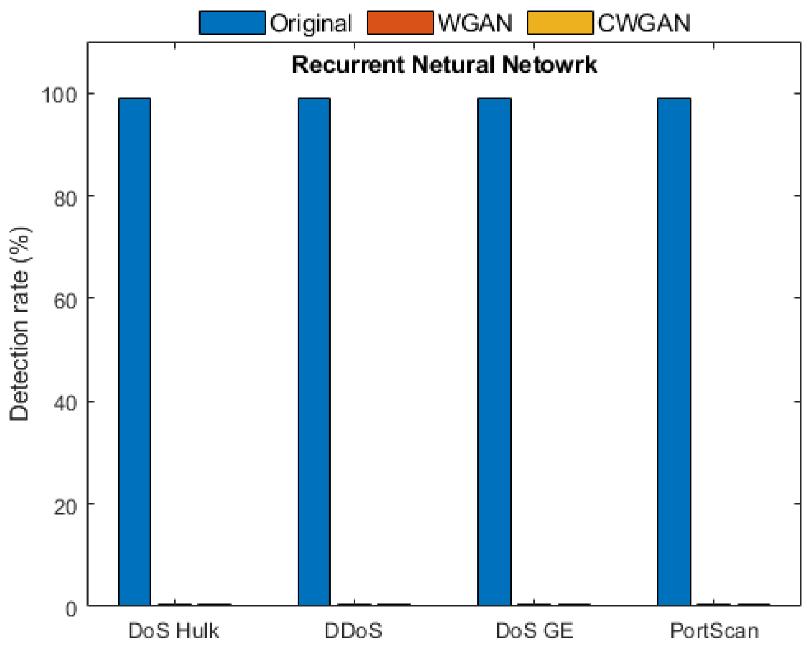


Figure 4.15: The detection rate of the Recurrent Neural Network with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that though the RNN has a more complex architecture than selected ML algorithms, it did not show better performance in defending against adversarial attacks.

by modifying the non-functional features solely.

The EIR (%) values of the eight algorithms for adversarial datasets of the four attack types are shown in table 4.5. By checking the EIR values, it is noticed that for the machine learning algorithms, the NB, LR, and RF do not have the capability to defend the adversarial attack at all (i.e., one hundred or nearly one hundred EIR values in both WGAN and CWGAN cases). Both the GB and the DT algorithms have a strong capability to defend against the adversarial attack of the DoS Golden-Eye. The DT algorithm also performs well in defending the DoS Hulk attack under some scenarios (i.e., note the huge change of the EIR values in the two cases). Regarding the deep learning algorithms, the CNN generally outperformed all the other two algorithms under adversarial attacks (i.e., a relatively low EIR value for both WGAN and CWGAN framework). The poor performance of RNN in detecting the adversarial samples reminds us that using more complex algorithms does not ensure the enhancement of the capability of the IDSs against adversarial attacks (comparing to simple ML algorithms). The DT, MLP, and CNN should be focused on in future

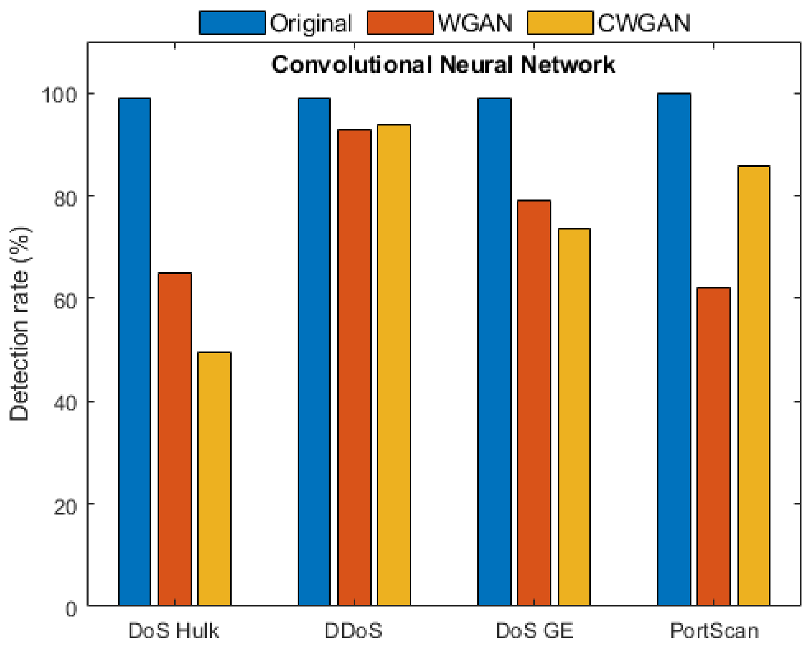


Figure 4.16: The detection rate of the Convolutional Neural Network with the original dataset and two adversarial datasets generated by the WGAN and the CWGAN. Note that the CNN has a consistent performance when defending the adversarial attacks from the WGAN and the CWGAN framework, and also it has a good ability to detect adversarial samples from all types of attacks.
””

studies regarding how to defend against adversarial attacks from GANs.

The performance of the WGAN and CWGAN on deceiving IDSs is close except for the DT algorithm in the DoS Hulk case. Regarding the training time, as we only feed one type of attack records into WGAN but all types of attack records into CWGAN each time, the training speed of a WGAN is much faster. However, as we need to train eight WGAN-based framework to match the function of a single CWGAN-based framework, the total training time is comparable between eight WGAN-based framework and one CWGAN-based framework.

4.3.4 Results Summary and Discussion

Regarding intrusion detection, tree-based algorithms provide the most promising results, and RF algorithm has the best performance among all eight algorithms. The

Table 4.5: The EIR(%) of the IDSs for different types of adversarial samples generated by the WGAN and CWGAN.

Attack	**Algorithm**	**NB**	**LR**	**DT**	**RF**	**GB**	**MLP**	**RNN**	**CNN**
DoS Hulk	WGAN	100	100	0.8	100	77.48	97.46	100	34.34
	CWGAN	100	100	94.79	100	79.18	90.04	100	49.95
DDoS	WGAN	100	100	81.28	100	86.89	7.2	100	6.06
	CWGAN	100	100	89.89	100	82.88	7.0	100	5.15
DoS GE	WGAN	100	100	0.1	100	0.1	14.53	100	20.53
	CWGAN	100	100	0.1	100	0.1	28.96	100	25.48
PortScan	WGAN	100	100	94.49	100	95.1	100	100	38.05
	CWGAN	100	100	96.71	100	96.7	99.59	100	14.01

deep learning algorithms are slightly underperformed than the tree-based ML algorithms. The outcome among the three DL algorithms are comparable. The Navie Bayes and Logistic Regression algorithms are not able to capture the malicious records very well, and should not be used to build up an IDS.

The results in the last section also show that the performance between the WGAN and the CWGAN is consistent in deceiving the IDSs. Among all the selected ML/DL schemes, the CNN-based IDS is the most robust under adversarial attack. There are two possible reasons that CNN has better resistance against adversarial attacks. On the one hand, the discriminator may not able to perfectly mimic the performance of the CNN, and hence, fails to provide high-quality feedback to the generator. Recall that we use an MLP for the discriminator, which probably can not capture the feature extraction process (by convolutional layers) of the CNN. On the other hand, the generator used in this study may not have sufficient capability (i.e, need more neurons and layers, or use more sophisticated architectures other than MLP) to generate the required adversarial samples. Unfortunately, at this juncture, there is no promised method regarding the explainability of the GANs. The aforementioned point will be investigated in future work.

The WGAN and CWAGN frameworks in the present study are designed for black-box attacks. However, the proposed frameworks can also be used for white-box attacks as well. In white-box attack scenarios, the internal parameters and gradients of the

IDS will be known, and hence, we can investigate which non-functional features contribute more to the final prediction of the ML/DL algorithms. This can be done by using some machine learning interpretation software such as SHAP (SHapley Additive exPlanantions). Subsequently, we can examine the internal gradients of the generator and check its parameter updates. We can further freeze the non-functional features with minor contribution and tune up the GAN to focus on modifying the significant non-functional features. In such a way, both training efficiency and framework capability should be enhanced.

Intrusion prevention system (IPS) is an active network protection system. Similar to IDS, it monitors network traffic and attempts to identify potential threats using signature, anomaly, or hybrid detection methods. However, unlike an IDS, an IPS takes action to block or remediate an identified malicious activity. While an IPS raises an alert, it also helps to prevent intrusion from occurring. IPSs are ideal for environments where any intrusion could cause significant damage, such as intrusion of bank databases or military private networks. The ML/DL algorithms have also been used to build up IPSs [12]. Thus, by replacing an ML/DL-based IDS with an ML/DL-based IPS, the present frameworks can also be used to test the robustness of the IPS. The general process is exactly the same.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Over the past years, the ML and DL algorithms have been employed to construct effective IDSs. In this thesis, we propose a WGAN-based framework to generate adversarial samples that can deceive ML/DL-based IDSs. The proposed framework only include restricted modification operations and the output in the framework is carefully regulated, which enables the frame to evade ML/DL-based IDS while preserving the nature of the target malicious attack. In addition, we present a variant of the proposed framework, which is based on Conditional WGANs (CWGANs). The variant framework can be used to generate adversarial traffic with the same level of evading rate. However, the training phase of the variant framework is significantly simpler because the variant framework can be trained for multiple types of malicious attacks, such as DDoS and port scan, simultaneously. In our research, we validated the effectiveness of the proposed framework by comparing the detection rate of the original dataset and that of the generated datasets. Our experimental results indicate that the proposed framework can completely deceive the IDS based on NB, LR, RF, or RNN. Also, the framework can partially evade the IDS based on DT, GB, or MLP.

5.2 Future Work

5.2.1 Other Deep-generative algorithms

The generator and the discriminator in the GAN are designed for two different tasks, namely, the generative task and the discriminative task. The generative task is usually more challenge than the discriminative one as the discriminator only needs to judge if an input record is real or fake (generated) instead of creating a new sample which is expected to be similar to the input one. Hence, designing a more sophisticated generator is more helpful in generating data of expected distribution (i.e., generate adversarial samples to attack IDSs). One of the good candidates for the generator is the Long Short Term Memory (LSTM), and an illustration of the architecture is shown in Fig. 5.1. In an LSTM, instead of generating all the features at the same time, the LSTM produces each feature one by one when obtaining information from a previous LSTM cell. Meanwhile, the attention mechanism can be added to each LSTM cell to simulate the correlations among adjacent features. In addition, at the output layer, a pre-defined statistical model (i.e., Gaussian mixture model) can be used to further mimic the distribution of each feature in the dataset. In a word, the LSTM can not only craft samples but also control the distribution of the generated samples based on the original dataset distribution and relationships between adjacent features.

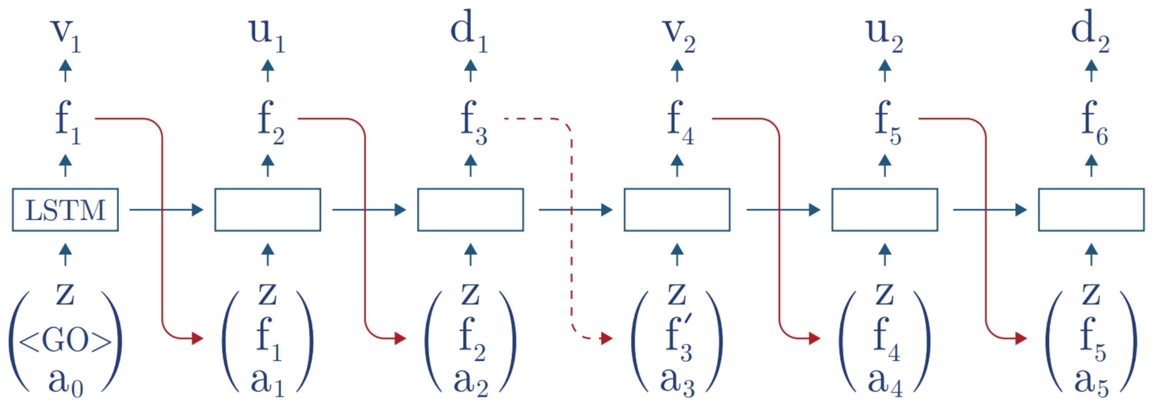


Figure 5.1: A schematic illustration of a Long Short Memory Term network used as a generator [59].

Another very good candidate is a variational Autoendoer (VAE) [58]. Similar to a regular Autoencoder, there are two sub-components in a VAE, an encoder and a decoder. However, instead of encoding the original input data into a single data point/vector, the VAE encodes the input data into a distribution, as shown in Fig. 5.2. The encoded distributions are usually chosen to be standard normal so that the encoder can be trained to return the mean and the covariance matrix that describes these Gaussian (the assumed distribution). The reason why an input is encoded as a distribution instead of a single point/vector is that it makes the VAE possible to express the latent space regularisation in a natural way: the distributions returned by the encoder are enforced to be close to a standard normal distribution [36]. The VAE has been proven to have better performance than MLP on tabular data generation [2].

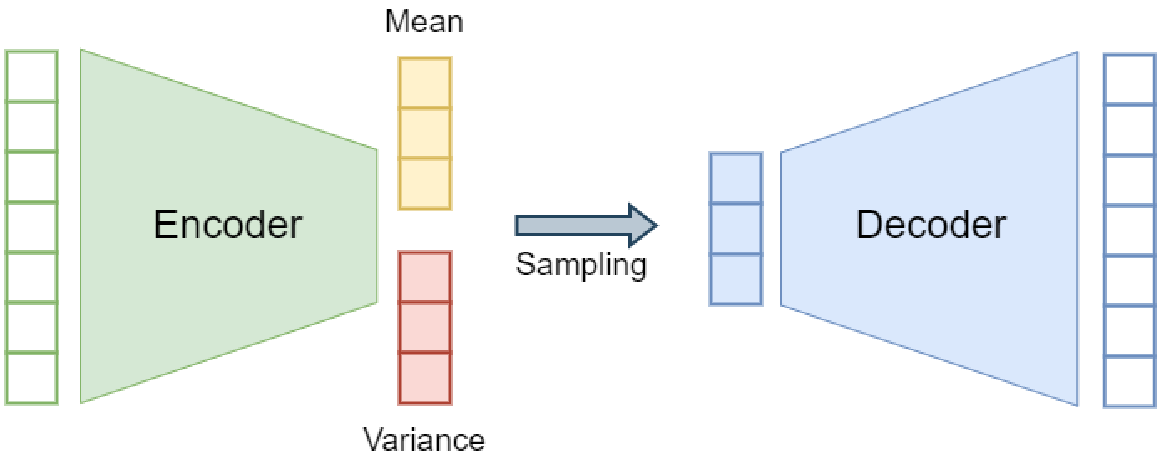


Figure 5.2: A schematic illustration of a variational Autoencoder [16]. Note that the input data is encoded into a distribution instead of single point/vector.

The above two mentioned architectures will be implemented and tested soon in a future study. A thorough comparison will be made to examine if the LSTM and VAE can get better results on adversarial sample generation (i.e., comparing the resulted detection rate and evasion increase rate).

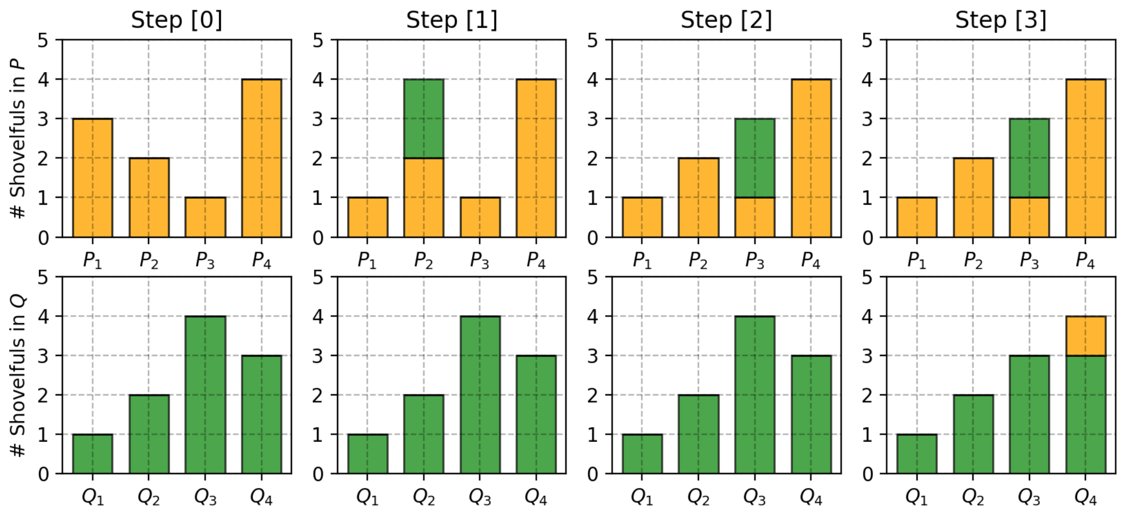


Figure 5.3: A schematic illustration of the Earth Mover Distance between two one-dimensional distribution [57].

5.2.2 Dissimilarity Measurement

One important aspect is to evaluate the dissimilarity between the original traffic records and the generated traffic records. It is expected that the adversarial records can evade the detection of IDSs but meanwhile have a higher similarity to the original records. The Earth Mover Distance (EMD) [2] is a widely used method to evaluate the dissimilarity between the dissimilarity of two distributions. An example of how to calculate the EMD between two distributions is shown in Fig. 5.3. Two distinct data distributions p and q are given. Subsequently, the two datasets were manipulated in several steps to make them identical. The moving distance with the minimum effort required to complete the manipulation is called EMD. Note that there is an infinite number of ways to manipulate the data to obtain two identical distributions. However, only the distance of the one with the minimum effort is called the EMD. In addition, in the example, the modification of samples comes from both datasets, but this is not the case in our investigation (i.e., we keep the original traffic records unchanged).

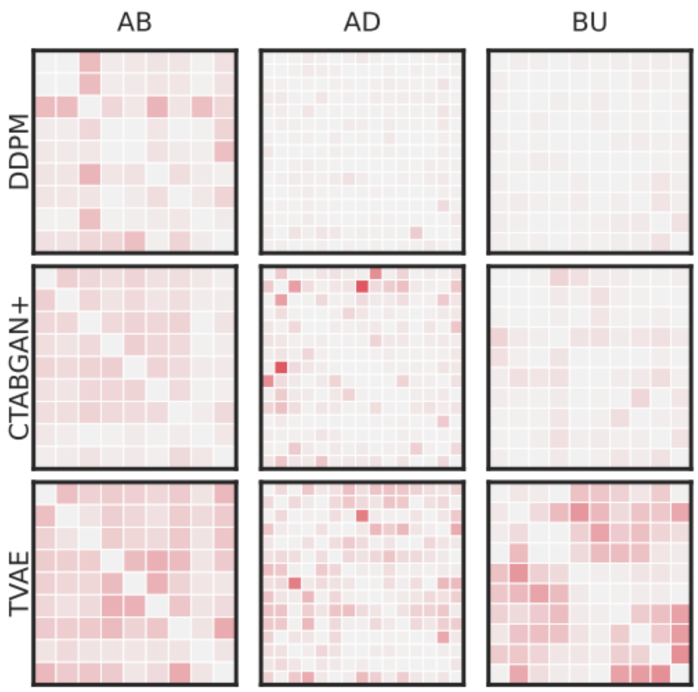


Figure 5.4: The absolute difference between correlation matrices computed on real and synthetic datasets [6]. A more intensive red colour indicates a higher difference between the real and synthetic correlation values.

The given example shows how to calculate EMD for two one-dimensional distributions. However, the input network security dataset for our investigation is two-dimensional (i.e., there are more than 600 thousand data entries in the dataset and each row has more than 80 features). Hence, a more complex way to evaluate the two-dimensional EMD is required (i.e., an average EMD based on the EMD between each column of two datasets) [3]. Note that having the same EMD does not mean the two sets of generated adversarial records are very similar, but they indicate that similar efforts are required to convert them into related original traffic records.

Another popular metric to evaluate the dissimilarity of the tabular dataset is the correlation matrix, which composes of the correlations between two column data [7]. Akim et al. developed a framework to synthesize the tabular data using a diffusion model [6]. They calculate the absolute difference between correlation matrices of the original and generated dataset as shown in Fig. 5.4. If the generated dataset maintains the correlation among features as the original dataset, the absolute difference should be small (light red colour in Fig. 5.4), which is what we expected.

5.2.3 Federated Learning and GAN

Federated learning is a type of distributed machine learning approach where the goal is to train a centralized algorithm while the training data remains distributed over a large number of decentralized edge devices or servers. A schematic illustration of the federated learning technique is shown in Fig. 5.5. In Fig. 5.5, the algorithm on each local device is first trained with the local dataset, and subsequently, the updated parameters and training gradients of the local algorithms are encrypted and sent to a central algorithm (on the central server). The central algorithm synthesizes this information and updates its parameters and gradients. The central algorithm also provides feedback to local algorithms for the next step of training. The aforementioned procedure keeps on going until the test performance of the central algorithm reach expectation. Federated learning has been widely used in the network security community. There are two major advantages of federated learning. On the one hand, as the central algorithm has no access to the local datasets, federated learning can address some critical issues, such as data security and privacy-preserving, in algorithm training. On the other hand, with federated learning, a large dataset can be broken up into several small datasets which are used to train multiple local algorithms, and the central algorithm is updated based on the factors of the local algorithms; thus, the training efficiency can be significantly increased with federated learning.

The federated learning technique has been used with GANs. Rasouli et al. [45] proposed a Federated Generative Adversarial Networks (FedGAN) for training a GAN across multiple distributed data resources subject to communication and privacy constraints. The FedGAN was tested on multiple image data (MNIST, CIFAR-10, and CelebA), and time series data (household electricity consumption and EV charging sessions) to study its convergence and performance. Their results showed that FedGAN has similar performance to general distributed GAN, while reduced communication complexity. Regarding the present study, federated learning could also be used in the proposed WGAN framework. First, as we increase the number of attack types considered, the size of the training dataset may grow dramatically. Thus, training a centralized WGAN framework may be no longer efficient. Second, it might be very dangerous to release certain intrusion detection datasets to the public. With federated learning, the WGAN framework can be trained without access to the dataset. These points will be investigated in future work.

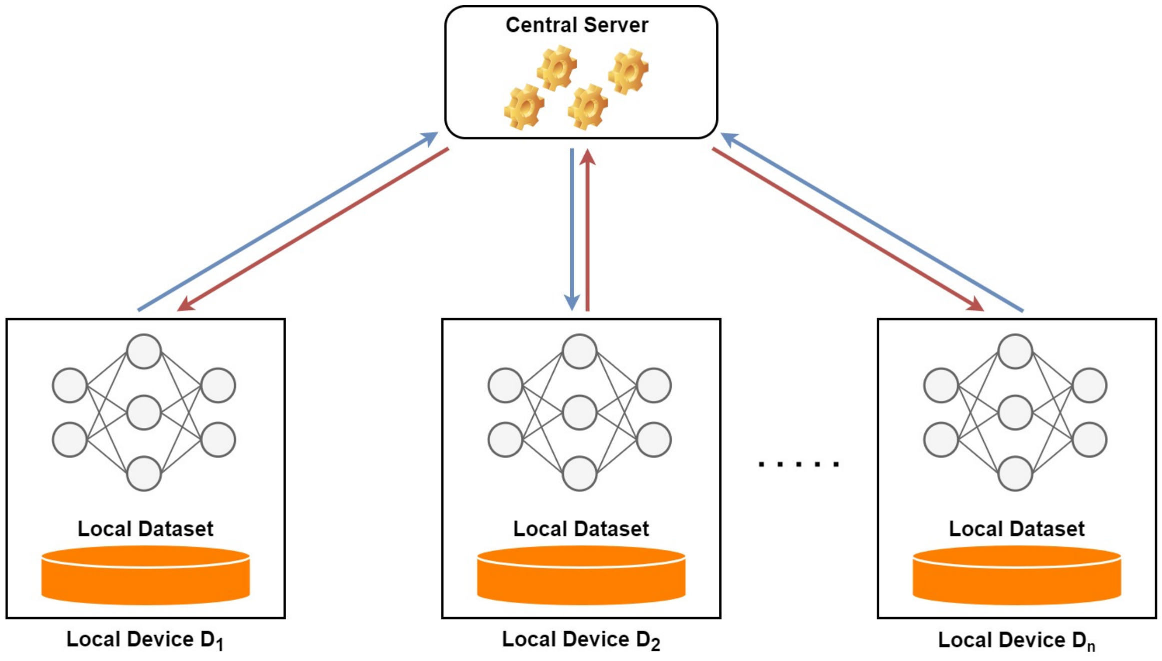


Figure 5.5: A schematic illustration of the federated learning technique.

5.2.4 GAN-based Data Synthesis for IDS training

One of the major challenges in training ML/DL-based IDSs is that the training datasets are usually highly imbalanced. The performance of the IDSs declines significantly in the case of learning from imbalanced data. Synthetic data generation is one of the efficient solutions to enhance the training of the ML/DL-based IDSs with an imbalanced dataset. The synthetic minority classes can adjust the class distribution of the original dataset and overcome the bias of the algorithm training. GANs and their variants have been widely used for tabular data generation [5] [59] [61]. However, only a limited number of them focus on data synthesis to enhance the training of IDSs. Lee and Park [30] employs a vanilla GAN to generate virtual data similar to the existing data to solve data imbalance. The CICIDS2017 dataset was oversampled using the GAN, and subsequently, the new dataset was fed into a Random Forest algorithm for classification. Their results show that the performance of the RF algorithm is significantly improved on multiple minority attack classes (i.e., Heartbleed and Bot) with the oversampled dataset.

Our proposed WGAN and CWGAN framework can be slightly modified (e.g., remove the pre-trained IDS) to accommodate data synthesis tasks. The results from [30] can be used as a benchmark to evaluate the capability of the proposed frameworks. Note that data synthesis tasks have more requirements on the output of the generator. It is usually expected the generated data does not significantly impact the statistical distribution of the input data, and also the virtual data remains a similar correlation among data features. This is a promising future direction and should be investigated.

Bibliography

[1] Preeti Aggarwal and Sudhir Kumar Sharma. Analysis of kdd dataset attributes - class wise for intrusion detection. *Procedia Computer Science*, 57:842–851, 2015. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

[2] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 324–330, 2009.

[3] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *SODA*, volume 8, pages 343–352, 2008.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Karim Armanious, Chenming Jiang, Marc Fischer, Thomas Küstner, Tobias Hepp, Konstantin Nikolaou, Sergios Gatidis, and Bin Yang. MedGAN: Medical image translation using GANs. *Computerized Medical Imaging and Graphics*, 79:101684, jan 2020.

[6] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey, 2021.

[7] C. J. Brien, A. T. James, and W. N. Venables. An analysis of correlation matrices: Variables cross-classified by two factors. *Biometrika*, 75(3):469–476, 1988.

[8] Jason Brownlee. *Generative adversarial networks with python: deep learning generative models for image synthesis and image translation.* Machine Learning Mastery, 2019.

[9] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 3–14, New York, NY, USA, 2017. Association for Computing Machinery.

[10] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. Adversarial attacks for tabular data: Application to fraud detection and imbalanced data, 2021.

[11] Sumouli Choudhury and Anirban Bhowal. Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In *2015*

International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pages 89–95, 2015.

[12] Soubhik Das and Manisha J. Nene. A survey on types of machine learning techniques in intrusion prevention systems. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2296–2299, 2017.

[13] D.E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987.

[14] Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid, and Robert Abbas. Effective intrusion detection system using xgboost. *Information*, 9(7), 2018.

[15] Yalei Ding and Yuqing Zhai. Intrusion detection system for nsl-kdd dataset using convolutional neural networks. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, CSAI '18, page 81–85, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Carl Doersch. Tutorial on variational autoencoders, 2016.

[17] Phan The Duy, Le Khac Tien, Nghi Hoang Khoa, Do Thi Thu Hien, Anh Gia-Tuan Nguyen, and Van-Hau Pham. Digfupas: Deceive ids with gan and function-preserving on adversarial samples in sdn-enabled networks. *Computers & Security*, 109:102367, 2021.

[18] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. *A Geometric Framework for Unsupervised Anomaly Detection*, pages 77–101. Springer US, Boston, MA, 2002.

[19] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1):18–28, 2009.

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[21] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[22] He Huang, Philip S. Yu, and Changhu Wang. An introduction to image synthesis with generative adversarial nets. *CoRR*, abs/1803.04469, 2018.

[23] K. Ilgun. Ustat: a real-time intrusion detection system for unix. In *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 16–28, 1993.

[24] Philokypros Ioulianou, Vassilios Vassilakis, and Ioannis Moscholios. A signature-based intrusion detection system for the internet of things. 07 2018.

[25] ShengYi Jiang, Xiaoyu Song, Hui Wang, Jian-Jun Han, and Qing-Hua Li. A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters*, 27(7):802–810, 2006.

[26] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[28] Christopher Kruegel and Thomas Toth. Using decision trees to improve signature-based intrusion detection. In Giovanni Vigna, Christopher Kruegel, and Erland Jonsson, editors, *Recent Advances in Intrusion Detection*, pages 173–191, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] JooHwa Lee and KeeHyun Park. Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, 25, 02 2021.

[31] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Volume 38*, ACSC '05, page 333–342, AUS, 2005. Australian Computer Society, Inc.

[32] Wenjuan Li, Steven Tug, Weizhi Meng, and Yu Wang. Designing collaborative blockchained signature-based intrusion detection in iot environments. *Future Generation Computer Systems*, 96:481–489, 2019.

[33] Yang Li and Li Guo. An active learning based tcm-knn algorithm for supervised network intrusion detection. *Computers & Security*, 26(7):459–467, 2007.

[34] Jianxin Lin, Yingce Xia, Tao Qin, Zhibo Chen, and Tie-Yan Liu. Conditional image-to-image translation, 2018.

[35] Zilong Lin, Yong Shi, and Zhi Xue. IDSGAN: Generative adversarial networks for attack generation against intrusion detection. In *Advances in Knowledge Discovery and Data Mining*, pages 79–91. Springer International Publishing, 2022.

[36] Jingmei Liu, Yuanbo Gao, and Fengjie Hu. A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm. *Computers & Security*, 106:102289, 2021.

[37] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110:107332, feb 2021.

[38] Matthew V. Mahoney and Philip K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In Giovanni Vigna, Christopher Kruegel, and Erland Jonsson, editors, *Recent Advances in Intrusion Detection*, pages 220–237, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[39] John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, nov 2000.

[40] John McHugh, Alan Christie, and Julia Allen. Defending yourself: The role of intrusion detection systems. *IEEE Software*, 17(5):42–51, 2000.

[41] Souhail Meftah, Tajjeeddine Rachidi, and Nasser Assem. Network based intrusion detection using the unsw-nb15 dataset. *International Journal of Computing and Digital Systems*, 8(5):478–487, 2019.

[42] Yu-Xin Meng. The practice on using machine learning for network anomaly intrusion detection. In *2011 International Conference on Machine Learning and Cybernetics*, volume 2, pages 576–581, 2011.

[43] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[44] Leonid Portnoy, Eleazar Eskin, and Salvatore Stolfo. Intrusion detection with unlabeled data using clustering. 11 2001.

[45] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data, 2020.

[46] Paulo Angelo Alves Resende and André Costa Drummond. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.*, 51(3), may 2018.

[47] Maria Rigaki and Ahmed Elragal. Adversarial deep learning against intrusion detection classifiers. 10 2017.

[48] Saharon Rosset and Aron Inger. Kdd-cup 99: Knowledge discovery in a charitable organization's donor database. *SIGKDD Explor. Newsl.*, 1(2):85–90, jan 2000.

[49] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, 2018.

[50] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. Deep recurrent neural network for intrusion detection in sdn-based networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 202–206, 2018.

[51] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, page 53–58. IEEE Press, 2009.

[52] Hoang Thanh-Tung and Truyen Tran. On catastrophic forgetting and mode collapse in generative adversarial networks, 2018.

[53] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[54] Mueen Uddin, Azizah Abdul Rahman, Naeem Uddin, Jamshed Memon, and Suhail Kazi. Signature-based multi-layer distributed intrusion detection system using mobile agents. *International Journal of Network Security*, 15:79–87, 01 2013.

[55] Muhammad Usama, Muhammad Asim, Siddique Latif, Junaid Qadir, and Ala-Al-Fuqaha. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 78–83, 2019.

[56] Zheng Wang. Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6:38367–38384, 2018.

[57] Lilian Weng. From gan to wgan, 2019.

[58] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. *Modeling Tabular Data Using Conditional GAN*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[59] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *CoRR*, abs/1811.11264, 2018.

[60] Jiahai Yang, Peng Ning, X. Sean Wang, and Sushil Jajodia. Cards: A distributed system for detecting coordinated attacks. In Sihan Qing and Jan H. P. Eloff, editors, *Information Security for Global Information Infrastructures*, pages 171–180, Boston, MA, 2000. Springer US.

[61] Zilong Zhao, Aditya Kunar, Hiek Van der Scheer, Robert Birke, and Lydia Y. Chen. Ctab-gan: Effective table data synthesizing, 2021.

[62] Ming Zheng, Tong Li, Rui Zhu, Yahui Tang, Mingjing Tang, Leilei Lin, and Zifei Ma. Conditional wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. *Information Sciences*, 512:1009–1023, 2020.