Evaluating Adversarial Learning on Different Types of Deep Learning-based Intrusion Detection Systems using min-max Optimization

by

Rana Abou Khamis

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of
Master of Applied Science

in

Information Technology

Carleton University
Ottawa, Ontario

The undersigned recommend to

the Faculty of Graduate Studies and Research

acceptance of the Thesis

Evaluating Adversarial Learning on Different Types of Deep Learning-based Intrusion Detection Systems using min-max Optimization

Submitted by **Rana Abou Khamis**

in partial fulfilment of the requirements for the degree of

Master of Applied Science

Dr. Ashraf Matrawy, Supervisor

Dr. Chris Joslin, School Director

Carleton University

2020

Abstract

Network security applications based deep neural networks, including intrusion detection systems, are effectively applied to reduce false alarm rates and improve detection accuracy and robustness against various network attacks and anomaly activities. With the rapid increase of using deep neural networks, various growing types of adversarial attacks to deceive them donate a severe challenge to override the vulnerabilities of deep neural networks and achieve a security guarantee. In this research, we focus on investigating the effectiveness of different adversarial attacks and robustness of deep learning-based Intrusion detection using different Neural networks, e.g., Artificial Neural Network (ANN), convolutional neural networks (CNN), recurrent neural networks (RNN). We utilize the min-max approach to formulate the problem of training robustness intrusion detection against adversarial samples using UNSW-NB15 and NSD-KDD. We structure an optimization framework by applying the max approach to generate persuasive adversarial samples that maximize loss. On the other side, we minimize the loss of the incorporated adversarial samples during the training time. With our experiments on multiple deep neural networks algorithms and two benchmark datasets, we demonstrate that defense using adversarial training based min-max approach increases the robustness of the network under the assumption of our threat model and five state-of-the-art adversarial attacks. *Keywords: Deep Neural Networks, Intrusion Detection System, Adversarial Samples.*

Acknowledgments

I want to express my deep and sincere gratitude to my supervisor Dr. Ashraf Matrawy, Professor, and the lead of the Next Generation Networks (NGN) research group at Carleton University for providing me with valuable guidance and encouragement and directing me in the right path to pursue my goal. I was a great privileged and honor to work and study under his guidance and extremely grateful for what he has offered me. I'm extending my heartfelt thanks to my labmate in the NGN Research Group: Olakunle Ibitoye for friendship and help throughout my study. Thank you for the project discussions, for the late hours we were working together before deadlines, and for sharing advice, knowledge, and time.I want to thank all Next Generation Networks (NGN) Research Group for the fun and time we spent sharing knowledge and experience. I cannot forget to sincerely thank Dr. Omair Shafiq for the thoughtful comments and recommendations in my first research paper. Also, I want to thank Dr. Ali Arya for supporting and directing me in my study. I'm thankful to the School of Information Technology and all its member's staff for all the thoughtful guidance. A special thanks for Erenia Hernandez Oliver for supporting me. I also want to take this opportunity to thanks Dr. Audrey Girouard for offering me an internship through the Collaborative Learning of Usability Experiences (CLUE) program.

To my friends, thank you for being there for me when I needed it.

I'm very thankful to my husband Khalid Hejazi and my kids, Omar and Jana, for their love, patience, understanding, prayers, and encouragement to complete this

work. Thank you for supporting me each day and never letting me lose hope. It is only the power of your love and your belief that it has made it possible for me to dream and make those dreams come true.

My special thanks to my parents: Ahmad Abou Khamis and Lina Kabra. I owe all good in my life to you. Thank you for your countless sacrifices that have opened many doors for me to grow and succeed. I hope I make you as proud as you make me. My brother Firas and my sister Yara thank you for all the love, support, and prayers you had shown through my study.

To my parents: Ahmad and Lina

To my husband: Khalid

To my kids: Omar and Jana

Contents

List of Tables

List of Figures

Abbreviations

ADAM	Adaptive Learning Rate Algorithm
ANN	Recurrent Neural Network
CNN	Convolutional Neural Network
DF	Deep Fool
DL	Deep Learning
DT	Decision Trees
FGSM	Fast Gradient Simple Method
FN	False Negative
FNR	False Negative Rate
FPR	False Positive Rate
GAN	Generative Adversarial Networks
GPU	Graphics Processing Unit
IDS	Intrusion Detection System
JSMA	Jacobian-based Saliency Map
KNN	K-Nearest Neighbor
LR	Logistic Regression
LSTM	Long Short-Term Memory networks
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naive Bayes

NB	Naive Bayesian
NN	Neural Network
PCA	Principal Component Analysis
PGD	Projected Gradient Descent
ReLU	Rectified Linear Unit(s)
RF	Random Forest
RFE	Recursive Feature Elimination
RNN	Recurrent Neural Network
SVM	Support Vector Machine

Chapter 1

Introduction

This chapter covers an overview of the research in adversarial attacks in deep learning-based intrusion detection systems and the defense technique used to carry out the research questions and contributions. Lastly, a summary of the thesis structure.

1.1 Overview

Computer applications have undoubtedly increased reliance on machine learning and, particularly deep learning has led to high-performance tasks for various fields such as anomaly detection, computer vision, image classification, and speech recognition. Security applications-based deep neural networks (DNN) like Intrusion Detection System (IDS), malware detection, and spam-filtering have become essential in designing data protection, classification, and prediction tasks. There are many challenges associate with improving performance: detection accuracy, reduce false alarms, and detect novel attacks in the Intrusion detection system (IDS) that play an essential role in protecting cybersecurity and computer network. These different types of tasks rely on the intelligence to build a model that typically classify and discriminate between normal and abnormal data, like attack and normal packets. Toward this direction, a majority of IDSs enhance their capabilities by using neural networks (NN) towards

deep learning to make better learning and processing for big data and detecting different and novel attacks for future prediction with high accuracy and low false alarms. However, the rapid increase of using DNNs and the volume of data traveling through systems, the compromise by remote intrusions, and sophistication of attack tools are increased. Therefore, various researches [6] [3] [7] [8] confirm that there are different growing types and methods of attacks that donates a severe challenge on vulnerabilities of DNN architecture design.

The fact that the training of DNNs is based on data, classification tasks can be carefully manipulated by crafted perturbations called adversarial samples. Adversarial samples are often visually imperceptible to reliably mislead a model toward incorrect classification and evade detection [9] [6]. Adversarial samples include emails, packet flow, malware, or virus files, are designed to avoid spam filters, intrusion detection systems (IDS), and malware detection. Adversarial attacks can attempt to attack data or models in a way to make the decision boundary between the regular and the crafted data shift inaccurate. Further, they are many types of adversarial attacks and algorithms to generate adversarial samples [6] [10] [11] [9] [12] [13]. The protection and robustness against adversarial attacks is a growing challenge [14] [8] and should always be guaranteed by DNN researchers and designers [15] [16] [17] [18].

Further, most of the work recognise adversarial attacks in the image classification domain [19] [12] [20] [21] and hardly any in intrusion detection systems. Toward covering the issue of securing DNNs against adversarial attacks, we study the effectiveness of the adversarial attacks against various state-of-the-art models in deep neural networks and their robustness in the field of the intrusion detection systems. In this research, our primary goal is to develop multiple resistance deep neural networks (ANN, CNN, and RNN) based intrusion detection systems and investigate their robustness against different state-of-the-art adversarial attacks that have been recognized in different classification domains, including image classification, text, speech

recognition, malware detection [21] [22] [23]. However, they have not been comprehensively utilized and studied their vulnerability in the Intrusion detection domain. We believe that deep neural networks, including CNN and RNN, will make a difference in increasing detection accuracy and reduce false alarm rates and training time in an adversarial environment. To do so, we utilize min-max formulation [21] [23] [22] that incorporates and augments crafted inputs during the training process. The min-max formulation is a combination of two problems. An outer minimization problem that minimizes the loss function during the learning phase, and an inner maximization problem that craft adversarial samples with maximum loss.

1.2 Research Questions

The thorough study of related work in adversarial attack and defense helped to derive several research questions:

1. How can we build resistant deep learning-based IDS against adversarial attacks?
2. Does our IDS framework based on min-max formulation improve the robustness against adversarial samples and minimize the risk over adversarial samples in all selected deep neural network architectures?
3. Do UNSW-NB15 and NSD-KDD datasets correspond differently under adversarial attacks?
4. Does min-max optimization represent a universal security cure for adversarial Intrusion detection trained by one type of adversarial attack method?

1.3 Thesis Contributions

There is a lack of experiments and proof of concept of a security guarantee against adversarial attacks in the field of Intrusion detection systems. After reviewing the literature in the field of adversarial attack against deep learning-based IDS, none of the current studies demonstrated the best defense techniques and optimization models among different adversarial attack methods to boost the robustness and optimize intrusion detection systems in a deep learning network.

This research aims to study the robustness of multiple deep neural networks (ANN, CNN, and RNN) in the Intrusion detection field through the aspect of robust attack and defense in an adversarial environment. We apply a min-max optimization to attack and defense the deep learning-based IDS. We utilize the IDS framework proposed in [1] to generate robust adversarial samples and evaluate their impact by inner maximizer and to incorporate them during the training phases and reduce the loss by the outer minimizer.

Several existing adversarial attack methods were proposed in the literature, we utilize the five most common state-of-the-art attack methods including Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), Projected Gradient Descent (PGD), Carlini and Wagner Attack(CW) and DeepFool to generate adversarial samples. CW adversaries are the most robust attacks in UNSW-NB15.The results are discussed in Chapter7.

We employ feature selection using Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA) to build the baseline models. RFE achieves noticeably better performance than PCA in terms of accuracy, recall, precision, F-score, and AUC. UNSW-NB15 and NSD-KDD networks achieve an accuracy of more than 95%. Chapters 5 and 6 have the details of experiments.

To the best of our knowledge, we are the first to demonstrate comprehensive study

and evaluation of min-max formulation on multiple deep neural networks(ANN, CNN, and RNN) and two datasets under the risk of adversaries from multiple state-of-the-art attack methods.

Our main contributions demonstrate that min-max optimization problem is manifestly beneficial in a different type of deep neural networks (ANN, CNN and RNN) in the field of the intrusion detection system. Our major contributions are elaborated below:

- Firstly, to the best of our knowledge, this is the first research elaborates comprehensive experimental study and analysis of min-max optimization that cast both attack and defense in deep learning-based Intrusion Detection System on three popular deep neural network architectures (ANN, CNN, RNN) and two IDS benchmark datasets in an adversarial environment. We provide a performance comparison between five state-of-the-art adversarial attacks, three DNNs architectures, and two datasets. The result provides evidence that the IDS framework proposed in AbouKhamis et al. [1] can reliably solve the optimization problem in deep learning-based Intrusion Detection Systems trained by a single type of adversarial attack. We are also the first to achieve this degree of robustness on UNSW-NB15 and NSD-KDD in an adversarial environment. Details in Chapter 7. Based on this analysis and the degree of robustness that we achieved, we have inspired our second contribution.

- Secondly, we investigate if the utility of the IDS framework proposed in AbouKhamis et al. [1] based on min-max formulation in deep learning-based IDS may consider a general defense against multiple types of adversarial attacks. We attack the adversarial IDS trained by a single type of adversarial via multiple adversarial attacks. The results show that adversarial IDS robustness

is substantially improved and attaining higher robustness compared to baseline IDS models trained by a clean dataset. We obtain 60 - 70% accuracy for both UNSW-NB15 and NSD-KDD for adversarial IDSs, while obtaining accuracy between 30-40% for UNSW-NB15 and NSD-KDD for baseline IDSs under multiple adversarial attacks. Evaluation of adversarial trained IDSs against multiple adversarial attacks is discussed in Chapter 8.

1.4 Published Papers

Throughout my studies in the program, I produced the following papers:

1.4.1 The Threat of Adversarial Attacks on Machine Learning in Network Security – A survey

Currently, we are preparing this survey paper [24] for submission and reviewing in a journal. This paper was a contribution of Olakunle Ibitoye, Rana Abou-Khamis, Dr. Ashraf Matrawy, and Dr. M. Omair Shafiq. My contribution in this paper is to establish a classification of machine learning in network security applications, survey adversarial attack methods in network security, and classify attacks based on network security taxonomy. The proposed adversarial threat model in [24] is partially used in this thesis in Chapter 3.

1.4.2 Investigating resistance of deep learning-based IDS against adversaries using min-max optimization

This paper [1] was published in IEEE ICC 2020 and a result of collaboration with Dr. Ashraf Matrawy and Dr. M. Omair Shafiq. This work was developed initially as a project for a Directed Studies course. The IDS framework based on the min-max approach proposed in this paper [1] is used in the current thesis in Chapter 6 and Chapter 7 to enhance the model resilience against adversarial attacks in deep learning-based IDS.

1.4.3 Evaluation of Adversarial Training on Different Types of Neural Networks in Deep Learning-based IDSs

This paper [2] is published in IEEE ISNCC 2020. This paper is a result of collaboration with Dr. Ashraf Matrawy. This paper is part of this thesis: Chapter 5, Chapter 6 and Chapter 7. The main purpose of this study is to investigate if the min-max approach in the adversarial training considers optimization solutions against adversarial attacks in different deep learning algorithms, including ANN, CNN, and RNN. We demonstrate that Algorithm 4.2 increases the robustness against five well-known adversarial attacks on two different IDS benchmark data sets: UNSW-NB15 and NSD-KDD.

1.5 Thesis Organization

We structure the remainder of the research as follows. Chapter 2 begins with background about Intrusion detection system, neural networks algorithms and deep learning. Chapter 3 is an overview of the classification of adversarial machine learning, adversarial crafting methods, and defense methods focusing on adversarial learning, followed by a literature review and a summary of some related work on adversarial machine learning, in particular in deep learning-based Intrusion detection. We mainly focus on adversarial methods and defense techniques studied by other researchers. Chapter 4 states our research problem and our methodology, including the IDS framework architecture and algorithm. In chapter 5, we explor Phase I experiments starting by describing the selected datasets and how we preprocess them using two feature selection techniques, followed by the architecture characteristics, learning setup, and evaluation metrics. Chapter 6 is about the performance of adversarial-free Deep learning-based IDS using two feature selection methods. Chapter 7 presents Phase II experimental results of attack and defense multiple deep neural networks using min-max formulation. In Chapter 8, we present Phase III results and evaluate the min-max formulation impact on multiple adversarial attacks. Finally, in chapter 9, we summarize our work, contributions, and the conclusion of our experiment results. Also, we present some recommendations on how we can strengthen our work in future research.

Chapter 2

Background

2.1 Intrusion Detection System

Cybersecurity has become essential in designing data protection and network integrity and usability. It authorizes and controls access to networks, endpoints, and applications to detect and prevent various threats and malicious activities from spreading in networks. Therefore, network systems should use various cybersecurity techniques and tools, including intrusion detection systems, malware detection, firewall, anti-viruses, and anti-spam to protect sensitive services and data from intruders and hackers.

An intrusion detection system (IDS) is a fundamental component in network systems to monitor and analyze real-time network flow and for any symptoms of suspicious, anomaly activities and issue alerts when intrusions are discovered. IDS detection methods are classified into two types: Knowledge-based (Signature-based) and Behavior-based (Anomaly-based). In the Knowledge-based (Signature-based), traffic is examined based on threat signature. This type requires continuous database update to not to fail and to identify new attacks. This type of detection has a low false alarm rate, but lack in the ability to detect unknown attacks [25]. While the Behavior-based (Anomaly-based), IDS inspects traffic based on the learned pattern

of activity behaviors that lead to recognizing unknown attacks. However, this type can be prone to higher false alarms [26].

One of the significant limitations of the standard intrusion detection system (IDS) is the urgency to filter and reduce false alarms [7] and the ability to detect unknown attacks. Toward this direction, the majority of IDSs enhance their capabilities by using machine learning-based systems, focusing on utilizing neural networks (NN) towards deep learning. Many machine learning algorithms were used to build IDSs including shallow and deep learning models including ANN, SVM [27] [28], KNN [29], Naive Bayes [30], Decision Tree. As a solution, Deep Neural Networks (DNN) was developed to make better learning and processing for big data and a variety of attacks for future prediction. We overview deep neural networks in the following section.

2.2 Deep Neural Networks Models

Deep Learning is a subclass of machine learning that is used to design more efficient models than shallow models in different industries. Most deep learning models are structured based on neural networks with multiple layers of linear models, including an input layer that receives the input information, hidden layers, and output layer that show the predicted result. This called Deep Neural Network (DNN), as shown in Figure 1. An increasing number of layers shifts the neural network from shallow to deep learning. The neural network works similarly to how the human brain is structured and function by using sophisticated algorithms and learning from massive data. Generally, deep neural networks do not require to memorize the previous prediction or output.

There are different types of Deep learning networks as shown in Figure 2 including Feed-forward (or Artificial) Neural Networks (ANN), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), and all of them are supervised

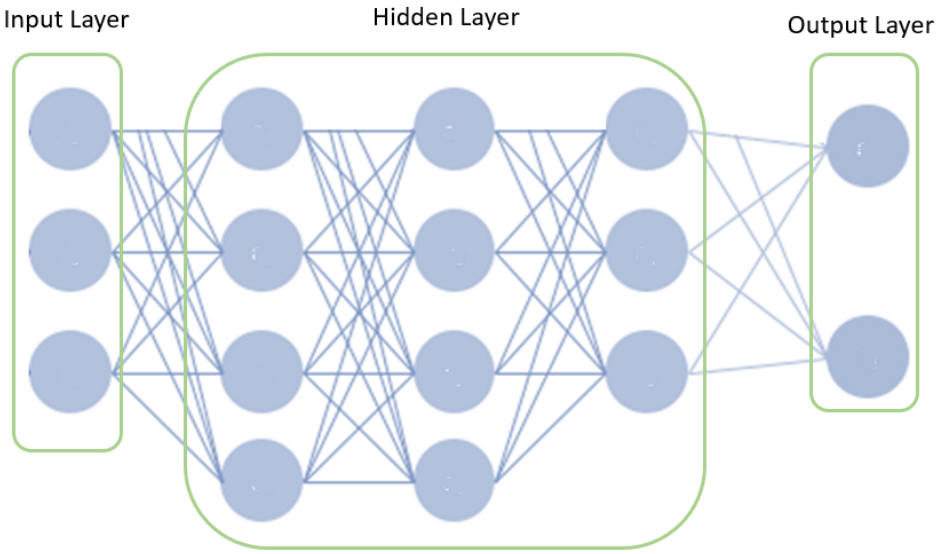


Figure 1: Deep Neural Networks Structure

learning models. The following section focuses on deep learning architectures and algorithms.

2.2.1 Deep Learning

DNN models learn by extract features from a large labeled dataset with high efficiency and accuracy [31]. Feature extraction is a time-consuming process in machine learning, while in deep learning is an automated process that can be done with less time. DNN efficiency is related to dataset size and massive computer power. By using GPUs for training with DNN models accelerate the learning process, while typical DNN could take a long time in the training, testing and optimizing deep models comparing to shallow models. The main priority in deep learning is a network architecture and network hyper-parameters.

The DNNs have a powerful capacity for fitting. However, DNNs have some potential problems that are controlled by the DNN capacity. Underfitting occurs when the network has less ability to learn the training dataset. While the overfitting occurs

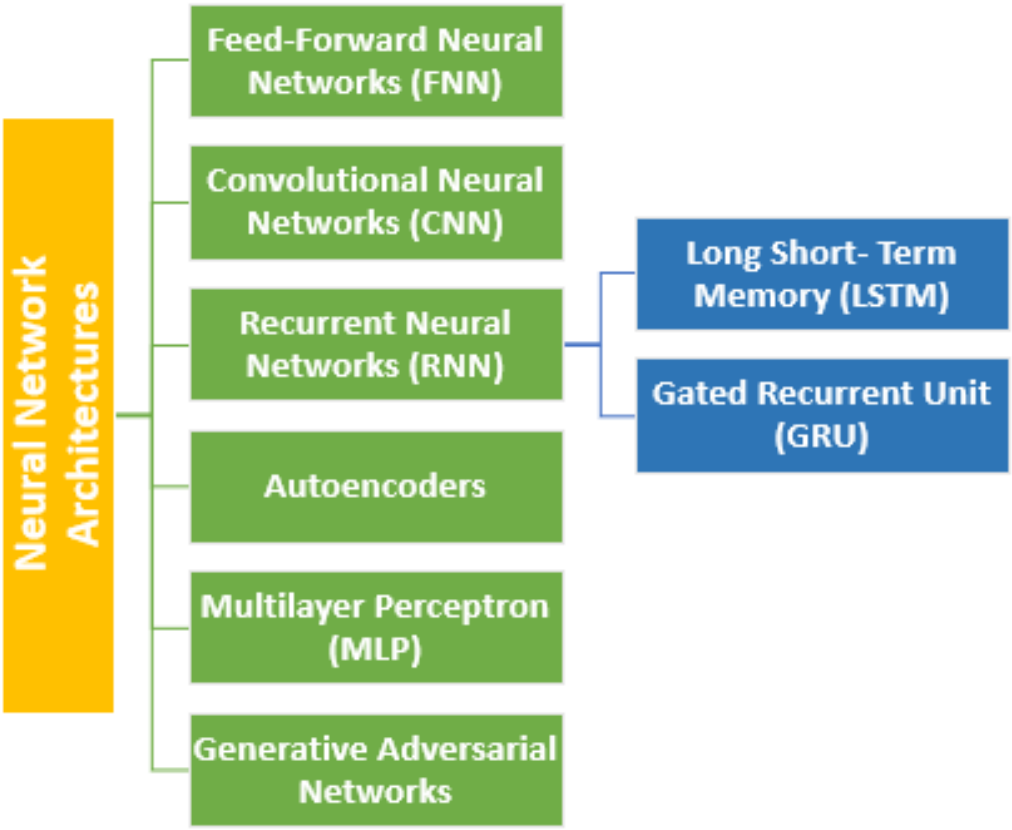


Figure 2: Type of Neural Network Architectures

when the capacity of DNN is large, and the model memorizes the training dataset instead of learning to predict and generalize unseen data sample [32]. For improving the generalization and avoiding overfitting, the network capacity should be large enough and use the regularization method. The number of nodes and layers control the DNN capacity [32]. In our deep learning-based IDS model, we use a binary classification $\mathcal{C}$ that minimize error when predicting *benign* and *attack* classes for new samples. The first *"Positive"* class indicates *attack instance* with *label=1*, while the second *"Negative"* class indicates benign instance with *label=0*. Therefore, the *False Positive (FP)* occurs if a benign instance classified as an attack and *False Negative (FN)* or *Evasion Rate* occurs if an attack instance classified as a benign.

There are various type of learning paradigms including supervised learning and unsupervised learning [24]. In a supervised classification problem, the DNNs are provided with pairs of input sample and output labels (x,y) $\in$ ($\mathcal{X}$,$\mathcal{Y}$) spaces from the training data $\mathcal{D} = (x_i, y_i)_i^m$ where m is number of instances. We define the detection classifier task over the input space $\mathcal{X}$ that consist of extracted n-dimensional features. Each network packet denote by $x = \{\mathrm{x}_1, x_2, ..., x_n\} \in \mathcal{X}$ where n is number of features. For example, our data features include x_1=scrip, x_2=sport, x_3=state,x_4=sbyte, etc. Also, we denote the labels by $y \in \mathcal{Y} = \{0,1\}$, where y=0 is benign packet, y=1 is an attack packet.

The DNN architecture, as seen in Figure 1, is made by many neurons connected to other neurons. The connection could be feed-forward or recurrent connections.The connections will be explained in the next section. Each neuron connection is called synapses and associated with weight $w_1, w_2, w_3..., w_n$ to multiply with input variables that also called features by the non-linear activation function. Weights are adjusted based on the input, the prediction output, and activation function. The bias value b is the value that shifts the activation function up or down. The activation function maps the output of neuron values depending on the function. The activation function has several options (e.g., Sigmoid, ReLU, Tanh, and Softmax). In our research, the activation function φ for the output is LogSoftMax and ReLU for the hidden layers. The Softmax function is the generalization of the Sigmoid that computes loss during training time and output values between 0 and 1. While ReLU linear function is a easy to compute and give fast learning, because it maps directly to one if positive or zero if negative. The last layer output gives the labeled result (e.g., "benign" or "attack") of the input traffic flow. Therefore, the DNN model should be trained to be applicable to map unseen examples. This phase is the most time consuming because of the complex computations required to complete the training phase. There are two procedures in DNN: forward and backward propagation. In forward propagation, the

initial state starts from the input layer, and it calculates the value of neurons in the first layer. For transferring the values to the next layers, activation function is used. The neuron values are transferred forward to the output layer.

To compare how far the prediction result from the target value $y \in \mathcal{Y}$ and to achieve an optimal model, we use a loss (or cost) function [3]:

$$\mathcal{T}(\emptyset, x, y)$$

Many loss functions (e.g., Mean Absolute Error (MAE), Mean Squared Error (MSE), cross-entropy) have a different effect in network training. Therefore, the network neurons keep adjusting the values of their parameters until achieving close prediction to the original value and reduce error.

The training process in DNNs is an iterative process to learn model parameters (weights and biases) $\emptyset \in \Theta$ that commonly modified using optimization "gradients" to achieve less error and optimize the loss function based on the learning rate ρ. The calculated error is used to adjust parameters $\emptyset$ until convergence to the minimum loss and achieve optimal parameters $\emptyset^*$ as Equation 1 based on [23].

$$\underbrace{\emptyset^* \in \underset{\emptyset \in \Theta}{\operatorname{argmin}} \mathcal{T}(\emptyset, x, y)}_{\text{outer minimization}} \tag{1}$$

Recently, various works [31] [33] [34] [35] proved that deep learning-based IDS effectiveness exceeds traditional ML-based IDS. Deep learning that uses modern hardware such as GPUs helps IDS to analyze better, detect, classify, and score anomaly behaviors and attacks for big labeled data and creating patterns. This helps in achieving high accuracy results that were not feasible before in classifying network traffic. However, the optimization of DNN to adversarial attacks is still a challenge for many researchers.

2.2.2 Artificial Neural Network (ANN)

ANN or known as a Feed-Forward Neural Network (FNN) process input in forwarding direction bypassing the input layer first, then the hidden layers, to the output layer. The main advantage of using ANN that it can map and learn any input to any output by using the activation function. The main task of ANN is regression and classification. One of the drawbacks of ANN that it cannot record sequential information. To overcome this problem, RNN and CNN architectures are used.

2.2.3 Convolutional Neural Networks (CNN)

CNN is a subdomain for deep neural networks that are inspired by human brain neural networks. Normally, CNN is used in the field of computer vision. However, many studies prove the success of CNN in many other applications, including network security applications like IDS, malware detection, and spam filtering [36] [37] [35]. CNN speeds the training process because it detects the significant and relevant information from the input.

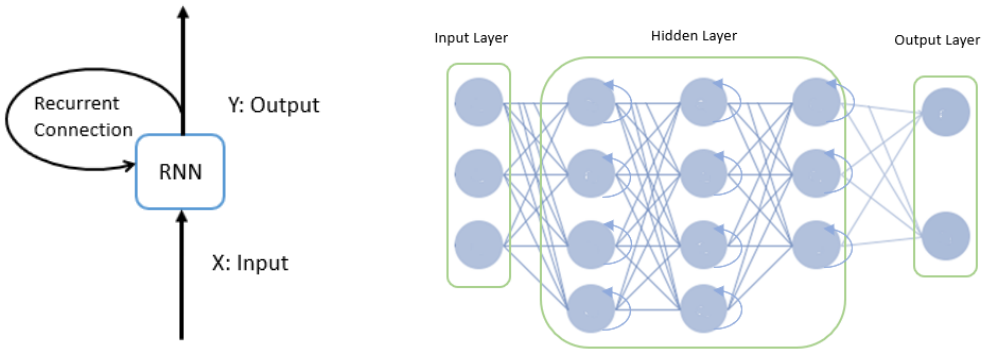
CNN architecture has various components, including convolution, pooling, and fully connected layers for sharing weight and pooling. The main layer is the convolution layer that applies a linear operation to extract input features and activation function on 3D vectors called feature maps. It is connected to the input neurons in the first hidden layer. Also, the Pooling layer normally comes after the convolutions layers, and it also collects and concise the information received from the convolution layer and reduces the dimension of the layer and parameter in training time. Therefore, the pooling layer reduces the computational time. The fully connected layer is the only layer that is fully connected that are used for classification. While in ANN, all neurons are fully connected with other neurons. The CNN layer flattens the

output and passes it to the output layer for label prediction using the softmax or sigmoid layer. Each layer in CNN are shaped by a set of many convolution kernels $W = w_1, w_2, w_3..., w_n$ and biases $B = b_1, b_2, b_3..., b_n$. The Convolution kernels with the input data, generate different feature maps. The main difference between ANN and CNN is that once the convolutional layer distinguishes features at a specific point in the input, it can recognize it again and no need to learn the same feature or pattern every time it appears the same input. The other essential advantage of the convolutional layers is learning the spatial pattern which leads to learning complex patterns. CNN learns simple features at the first layers (e.g., edges), and by the time it recognizes more complex features (e.g., faces), and at the final layer, it recognizes objects (e.g., stop sign).

CNN applied in different fields, including face recognition and image classification. In the domain of intrusion detection system, there are insufficient researches that invested in studying CNN in the domain of IDS [38] [37]. In this research, we applied CNN to build the IDS model for class detection and to utilize the feature learning in CNN.

2.2.4 Recurrent Neural Networks (RNN) and LSTM

RNN is a type of DNN that is a robust algorithm designed for sequential data. RNN differs from other deep learning architectures by its learning methods. It does not only have feed-forward connections, but it also has feed-back connections to the previous layers. Feed-back connection in RNN adds memory of past inputs where each unit receives the current and previous state where it feed back the out to it self. The output of previous steps goes as an input to the current step. It learns the relation between occurrences split by different times where these relations called long-term dependencies. The recurrent connection on the hidden state ensures the following information is recorded. It works similarly to the human brain, where it does not



(a) RNN (b) RNN

Figure 3: Recurrent Neural Networks (RNNs) Structure

start from scratch every time; it has a loop in its network that allows persisting information and share weight over time. It shares parameters across different steps. Therefore, the same weights are copied for the entire network input, and only output y and internal state u are changed. One of the main limitations of RNN is the limited-length sequences that cause a vanishing gradient problem, which leads to insufficient and inaccurate results. RNN has not able to link information with a large gap.

RNN has a capable architecture called Long Short-term Memory (LSTM). LSTM was introduced by [39]. It is used in various applications, including speech recognition in smartphones, translation, image classification. LSTM solves the standard RNN problem of handling long term dependencies and maintains the information for long term periods. RNN and LSTM are used in IDS in several works [36] [35] [40] to learn features that are associated with time. Packet flows that are from the same class has correspondence in attributes associated with time. We were inspired by RNN to create better models; we use LSTM in our work to build RNN-based IDS. Because deep learning has the potential to extract better representations from the data to create much better models, and inspired by recurrent neural networks, we have proposed a deep learning approach for an intrusion detection system using recurrent neural networks (RNN-IDS).

LSTM has different layers as shown in Figure **??**. The first layer called **Input Gate** layer x that decides what information we are going to update. The **Forget Gate** layer f is used to decide what information to keep and which information to forget. The decision of keeping or forgetting depends on the input x_t and the output of the previous state y_{t-1}. Also, the output and memory cell m are the other two layers. With those layers, LSTM remembers all information through time.

The following chapter focuses on adversarial samples in deep learning and how they are designed to deceive the models to make a mistake. In the next chapter, we review the adversarial threat model and classification and then explore various methods of adversarial generation and defense methods, followed by a summary of some related work on adversarial attack and defense.

Chapter 3

Adversarial Attack Classification and Threat Model

Adversarial attacks denote a challenge since deep neural network design is vulnerable to security attacks. Many recent works have focused on [41] [42] [14] [21] [43] [9] adversarial machine learning that study machine learning and in particular deep learning design and its vulnerabilities in resisting various attacks. Moreover, researchers mark the number of adversarial attacks that have been growing unexpectedly and study the capabilities and the limitations of adversarial attacks and the possible defensive methods. In this chapter, we present a background of adversarial samples and taxonomy of adversarial machine learning, followed by the methodologies of the state-of-art attack methods used to generate adversarial samples. Then, we focus on adversarial learning as one of the defense techniques as a countermeasure. At last, we review some related work in adversarial attacks and defenses.

3.1 Adversarial Samples

DNNs models are designed widely in various domains [44] [20] [45] [46] [9] which impetus for adversaries. Hence, DNN classifier performance is often failing by any

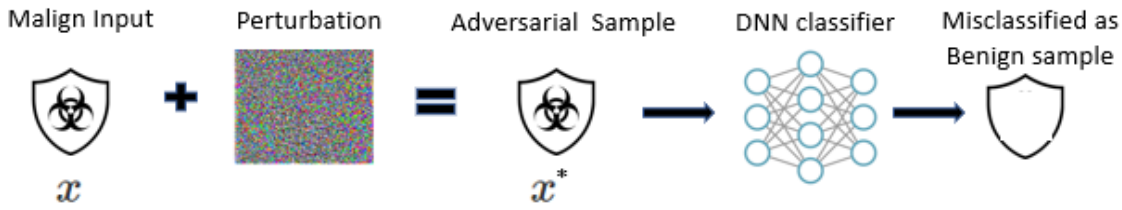


Figure 4: Adversarial sample

adversarial attempts to compromise the trained model [47] [48]. It is increasingly important to guarantee the protection and robustness against adversarial manipulation. Initially, Adversaries generated to fool image classification and deep neural network by Szegedy et al. [6]. Adversaries can evade classifiers by adding a calculated perturbation γ to legitimate samples $x \in \mathcal{Z}$(x) to create a new version $x^* \in \mathcal{Z}^*(x)$ called "Adversarial Sample" [6] [48], where $\mathcal{Z}^*(x) \subseteq \mathcal{Z}(x)$, $\mathcal{Z}$ is a set of allowed perturbations and $\mathcal{Z}^*$ is a set of adversarial samples that takes into consideration the maximum perturbations. This process is illustrated in Figure 4:

$$x^* = \overbrace{\underbrace{x}_{\text{original sample}} + \underbrace{\gamma}_{\text{perturbation}}}^{\text{adversarial sample}}$$

This leads the system to manipulate the decisions based on the adversary target. Steps for adding noise to the original sample are illustrated in Figure 4. In our work with continuous space, we focus on $\mathcal{L}_\infty$ to gives the max value of the parameters among each element samples: $||x||_\infty = max_i|x_i|$.

For instance, deep learning-based IDS can classify an input sample correctly as an attack with high accuracy can still be fooled by an attacker who targets this IDS classifier with a white-box attack to change the input traffic [48]. By adding a small adversarial noise that is calculated carefully, this classifier will recognize the new adversarial sample as benign traffic with high confidence.

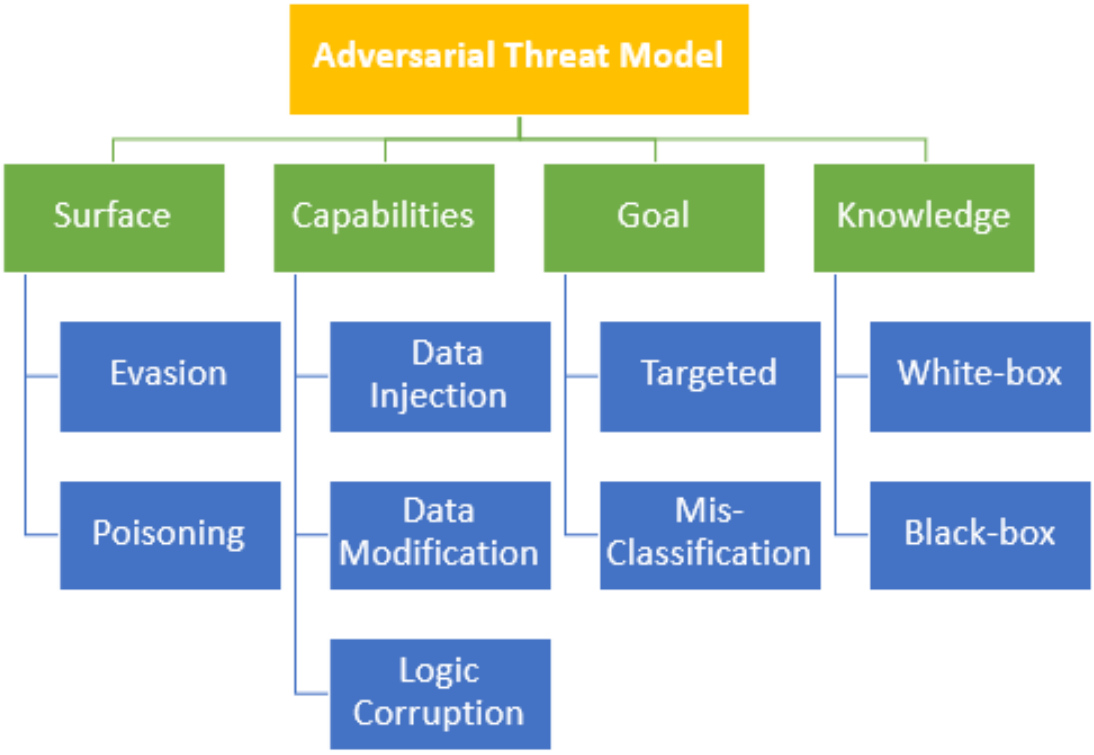


Figure 5: Adversarial Threat Model

3.2 Adversarial Threat Model

In the following section, we focus on the adversarial machine threat model in terms of capabilities, knowledge, goal, timing, and limitations of an adversary. Each adversarial attack can be classified under the following classifications, as shown in Figure 5. We design our adversarial attack taxonomy based on two taxonomies in [24] [49] surveys.

Adversarial Attack Surface

An adversary can attempt to manipulate either the data inputs or the processing of data to confuse the target model. The attack can be identified by the attack surface/time. Adversarial attacks can be classified based on the surface into two

attacks: poisoning and evasion attack [24].

- **Poisoning Attack:** This type of attack targets the machine learning algorithm by altering the training set during the training time. This attack is also called a causative attack [50] [12].

- **Evasion Attack:** This type occurs at the decision time and alters the test set during the testing or prediction time [51] [52] [7].

Adversarial Attack Capabilities

Adversarial capabilities determined by the amount of information available for the attacker about the system. Adversarial capabilities in machine learning can be determined in different phases: training and testing phase [49].

During the training phase, attacks attempt to corrupt how the model will learn by modifying training samples before training the target model. The adversary in the testing phase can fool the model to classify incorrect output. The effectiveness of adversarial attacks varies upon the amount of knowledge about the model. Modification for model or input might be implemented using different methods [49]:

- Data Modification occurs when the adversary has no access to the learning algorithm but has access to the entire training samples.

- Data Injection is a method that can be used when the adversary has no access to the training samples or the algorithm. However, it has ability to inject adversarial samples.

- Model Logic Corruption can be used when the adversary can control the model by adjusting the learning logic.

Adversarial Attack Goal

- **Targeted Attack:** The attacker confuses the trained model output with its desired target class (e.g., recognize spam email as a normal email in a spam filtering system [43]). This attack could also attempt to force specific input to produce a particular class (e.g., recognize stop sign as a yield sign)
- **Misclassification (or Reliability) Attack:** This occurs when the attacker has no specific target and only increases the prediction error [53].

Adversarial Attack Knowledge

Adversarial attack efficiency and limitation are specified by the degree of information available to the attacker about the model at the attack time. Adversary knowledge can be classified in general into White-box, or Black-box attacks [24].

- White-box Attack: Attacker has complete knowledge about the system, including the target model, algorithm, architecture, training input, parameters $\emptyset$, etc [10] [11].
- Black-box Attacks: In this type of attack, the attacker has no knowledge about the target model and employs information that may help in inspecting the weakness of the model [54].

3.3 Adversarial Samples Generation

As we recall in the previous section, adversarial attacks could be positioned during the training phase (poisoning attack) or prediction phase (evasion attack). To manipulate the training input or the learned classifier, small perturbations should be added to the original input to generate the adversarial samples. The robustness of the adversarial sample depends on the attacker's ability to find a sample relative to the original

input. In this section, we present well-known and state-of-the-art attack methods for generating adversarial samples that we utilize in our research experiments. Various other adversarial generating methods are reviewed in section 3.5.1.

Fast Gradient Sign Method (FGSM)

In [48], authors propose this method to save and ease the long computational time used for the old method "linear search" for creating an adversarial example. The main goal of this method is to move the DNN classifier's output to the new Target class and move the input toward this direction with a small noise [23]. As we denoted previously, the original input sample is x and the target class is y, and the loss function is: $\mathcal{T}(\emptyset, x, y)$. FGSM is all about updating the gradient of the loss function along with its sign direction in one step. The ∇ is used to calculate the derivative of the loss function. The derivative value of $\mathcal{T}(\emptyset, x, y)$ is used to find the slope of the function at a given point to adjust the parameters' values whether by increasing or decreasing them. This adjustment is a small value ρ and close to the loss function value that makes the perturbation not distinguishable The perturbation equation as in [48] is:

$$x^* = x + \rho sgn \nabla(\mathcal{T}(\emptyset, x, y))$$

Where ρ is the learning rate, and T is the loss function, x is the original sample, y is the target label. Thus, $x^* = x + \gamma$ is the adversarial example. This adversarial attack method can be computed using backpropagation. FGSM Adversarial examples are classified as a white-box attack because calculating gradient values required a knowledge of the DNN architecture.

Basic Iterative Method (BIM)

It is a way to apply the Fast gradient sign method (FGSM) multiple times with a small step size rather than applying it one time [47]. Repeatedly, with each training step, new adversarial samples will be created. Here is the perturbation equation:

$$x_0^* = x$$

$$x_{n+1}^* = x_n^* + \rho sgn\nabla(\mathcal{T}(\emptyset, x_n^*, y))$$

Projected Gradient Descent (PGD)

This type of adversarial attacks is more robust than FGSM [21]. It is a multi-step with a negative loss function. It overcomes the network overfit problem, and shortly comes of FGSM adversarial samples. It is more robust than FGSM, which utilizes the first-order network information, and it works well in large-scale constraints. In ℓ_∞-ball, PGD iterate to explore the maximum loss. The Resistance against PGD samples grants resistance against any first-order adversaries.

Carlini and Wagner (CW)

Carlini et al. [55] create attacks that outstanding the state-of-the-art and defeat defensive distillation using three distance metrics L_0, L_2, and L_∞ for generating adversarial samples. L_0 attack is not suitable for standard gradient descent. Therefore, they used the iterative algorithm to pick pixels that do not have much impact on the classifier decision and alter those pixels. After multiple iterations, a small subset is selected to generate an adversarial sample. While L_2 attack measures the distance between two coordinate x and $x^{'}$ and select a target class to find weights that encourage generating adversarial samples, that force them to misclassify the input to other classes with high confidence. L_∞ attack is used to measure the max changes

limit that is allowed to change between two coordinates. Standard gradient descent produces poor results. To solve this, they used iterative attacks, and they carefully choose a good constant called "warm start" to generate a fast adversary. The three adversarial attack algorithms defeat defensive distillation with a 100% success rate.

Deepfool

Moosavi et al. proposed [12] a DeepFool algorithm that can compute adversarial examples and defeat state-of-art classifiers to change output prediction label. They also indicate the generalizable concept, which means if adversarial examples created for a specific model can fool other models. They first assume the model is binary, and then extending the Deepfool method to a multi-class classifier. In each iteration, a minimal perturbation is computed based on equation four and Algorithm 1 in [12]. The algorithm stopped when the sign changed in the classifier. Deepfool attacks consider an efficient method to evaluate the robustness of classifiers.

3.4 Defense against Adversarial Attacks using Adversarial Learning

There are various approaches [56] [21] [57] to improve trained model robustness against adversarial samples. Mainly, the defense can be deployed by adjusting the classifier training process or eliminate adversarial perturbations from the dataset [58]. Adjusting the training process is feasible by adversarial training [6], Gradient Masking [16] or by defensive distillation [31]. While eliminating adversarial perturbations is feasible by detecting Adversarial Samples [59], Feature Reduction [60] or Generative Defense Adversarial Networks (GANs) [61] to clean input before passing to the classifier. Various other adversarial defense methods are reviewed in section 3.5.2.

Across various defense techniques against adversarial perturbation, we are interested in studying the adversarial learning (or training) approach in the intrusion detection system domain. The adversarial learning approach is about incorporating adversarial examples in the training phase. In [1], [23] and [21], authors successfully apply adversarial learning approach using min-max (or saddle-point) formulation and achieve robust model against packet [1] , malware [23] and images [21] adversarial attacks.

The model learned in Equation 1 in Chapter 2; the enhanced equation of [23] can be manipulated by adversaries γ that maximize the loss $\mathcal{T}$ as in Equation 2.

$$\underbrace{x^* \in \arg\max_{x^* \in \mathcal{Z}(x)} \mathcal{T}(\emptyset, x, y)}_{\text{inner maximization}} \tag{2}$$

We combined in Equation 3 between the inner maximization Equation 2); that aim to find an adversarial sample x^* from original sample x that achieves a high loss; and the outer minimization problem Equation 1; that aim to achieve optimal parameters $\emptyset^*$ and minimize the adversarial loss. This combination is the use of **min-max (or Saddle-point) formulation** [1] [21] [23]. The use of this approach allows us to achieve robustness against adversarial attacks.

$$\underbrace{\emptyset^* \in \underbrace{\arg\min_{\emptyset \in \Theta} \mathcal{T}(\emptyset, x, y)}_{\text{outer minimization}} \overbrace{[\max_{x^* \in \mathcal{Z}} \mathcal{T}(\emptyset, x^*, y)]}^{\text{inner maximization}}}_{\text{adv learning}} \tag{3}$$

3.5 Literature Review of Adversarial Attacks and Defense

We highlight the related work for adversarial machine learning, combined with adversarial attack methods and defense techniques against adversarial attacks. In particular, adversarial attacks in deep learning-based IDS. Researchers perform various studies of adversarial attack samples with their countermeasures and how they can easily deceive machine learning systems. More studies focus on taxonomy and classes of adversarial attacks, evaluating the resilience and robustness of the machine learning systems against those attacks. We overview some of the thriving researches in this field and highlight the methods, applications, algorithms, and datasets of their work and contributions.

3.5.1 Related work in Adversarial Attacks

Deep Learning-Based Intrusion Detection With Adversaries

In [3], authors studied the vulnerabilities of deep-learning-based IDS among adversarial attacks by evaluating the effectiveness of state-of-the-art attack algorithms using NSL-KDD dataset. While investigating, they show that generating adversarial examples can be affected by individual features, and they noticed that crafting many features is not practical. They compare different adversarial attacks: JSMA, RANDOM-TARGET, FGSM, DEEP FOOL, and CW attacks. Their experiment result shows that CW attacks have the lowest effectiveness comparing to other attacks, and this attack does not select features. At the same time, CA attacks show it is more robust to the state-of-the-art defenses. Moreover, generating adversarial examples using features in JSMA attacks is in balanced. They also show in their study that using adversarial examples targeting specific models can fool another model [3], especially

NN models, even if the different datasets trained it. This adversarial property called transferability. They prove that adversarial attack algorithms designed to attack image classifiers have different effectiveness to fool deep learning-based IDS. In their future work will focus on adversarial transferability. They evaluate the performance based on accuracy, Recall, precision, False Alarm, and F-score.

Evaluating Deep Learning-Based Network Intrusion Detection System in Adversarial Environment

[52] evaluated the proposed scalable deep learning-based ENIDS framework robustness in the adversarial environment against various attacks (MI-FGSM, L-BFGS, PGD, and SPSA) using NSD-KDD dataset. They compare their proposed deep learning-based NIDS with different well-known models, including SVM, RF, and LR, under adversarial attacks. They use different metrics to compare the model robustness, including accuracy(ACC), Precision Rate (PR), Recall Rate (RR), F-Sorce (FS), and Success Rate (SR). Their evaluation noticed that the DNN model under the MI-FGSM attack has the highest effectiveness among all models compared to other attacks where the accuracy of DNN decreased sharply from 76.8% to 30%. They noticed that deep learning-based NIDS performance decrease significantly.

Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks

[62] study adversarial samples effectiveness against deep learning-based Intrusion Detection System (IDS) within the context of an IoT network. The authors provide a comprehensive comparison between two deep learning models: a Self-normalizing Neural Network (SNN) and a Feed-forward Neural Network (FNN). They utilize and study input features normalization in a deep learning-based IDS in an adversarial environment. It increases the robustness of the deep learning model against various

adversarial attacks (FGSM, BIM, and PGD). In SNN IDS, the performance is higher with 9% than the FNN IDS, and the accuracy remains below 50% under adversarial attack. Normalization could not consider a suitable defense against adversarial attacks.

Online anomaly detection under adversarial impact

[63] study the effect of a poisoning attack of training data on IDS application with a finite sliding window. They study the poisoning attack with limited and full control of the training dataset using real HTTP traffic from a web server of Fraunhofer FIRST institute. This study shows that if the attacker has full control of the data, it is easy to attack while when applying additional constraints to have limited control of the training data. The attack fails. Therefore, adding those constraints adds protection approach against poising attack. Their results show that they cannot consider their method secure if the attacker has full control of the dataset.

Security evaluation of pattern classifiers under attack

[64] proposed a framework for empirical security evaluation that can be applied in different three real-life applications, including Intrusion detection system, spam filtering, Biometric Authentication. They propose an algorithm to sample training and testing sets. They evaluate their framework performance under causative adversarial attack using SVM and LR algorithm. The adversarial goal was set to maximize the percentage of a malicious testing sample as a normal sample by manipulating the training data. For IDS, they used a public data set of a webserver with 205 malicious samples collected in five days in 2006. Their result shows that the performance of the model under the causative attack remains similar to the performance without attack, and the performance starts to decrease when attack samples exceed 30% in the training set. Authors recommend the designer of classifiers to follow to use their

framework to evaluate the security of the classifier.

HopSkipJumpAttack: A Query-Efficient Decision-Based Attack

Chen et al. [65] developed a novel gradient direction estimation at the decision time using binary information as image and handwriting classification tasks. The HopSkipJump attack demonstrated efficiency over other state-of-art evasion attacks. Authors suggest using their novel attack to evaluate a new defense approach as the attach was evaluated against various existing defense techniques, including defensive distillation, region-based classification. The author conducted a comprehensive experiment using different data and models. They have used MNIST, CIFAR-10, CIFAR-100, and ImageNet. They also used different models, including convolutional networks, ResNet, and DenseNet.

The Limitations of Model Uncertainty in Adversarial Settings

Authors in [66] generate high-confidence-low-uncertainty (HCLU) adversarial examples that maximize the Gaussian process (GP) confidence and minimize the GP uncertainty. They work on the concept of transferability of adversarial samples to other algorithms. They implemented the HCLU adversarial examples on a Bayesian neural network (BNN). They have proved that Bayesian can be defeated by white-box and black-box attacks. Authors investigated GP uncertainty with adversarial examples using different classifiers, including BNN, DNN, and SVM, for different tasks such as handwriting digit recognition MNIST, malware, and spam. They used different adversarial attacks at test time like Carlini and Wagner and attacked targeting GP with high confidence and low uncertainty on the Gaussian process classifier.

Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning

Jang et al. jang2017objective design adversarial algorithm based on gradient-descent to produce useful adversarial samples for all images in a similar way of Deep Fool(NewtonFool) and reduce model confidence probability of correct class. They use new metrics: Canny edge detection, fast Fourier transform, and histogram of oriented gradients. They use these metrics to measure the effectiveness of adversarial samples in computer version applications using the MNIST and GTSRB dataset on the CNN model. They train a CNN classifier achieving 94.14% accuracy for the MNIST test set and 86.12% accuracy for the GTSRB test set. They compare their algorithm with FGSM, DeepFool, and JSMA. NewtonFool excels all results for all labels.

EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples

In this workchen2018ead, Chen et al. design an adversarial attack algorithm based on elastic-net regularization to craft samples based on the L1 norm for image problem and restoration and attack DNN. The elastic-net regularization is used for feature selection problems. Their experimental results on MNIST, CIFAR10, and ImageNet and achieve 99% attack success rate. They compare the effectiveness of their algorithm with CW, FGSM, and Iterative FGM using the success rate as an evaluation metric. Their attack EAD achieve 100% success rate in all datasets. Also, they show how their attack can also break defended DNN that use Defensive distillation. They show that EAD attack samples improved attach transferability. Finally, although they use adversarial training using EAD and CW on MNIST, the success rate remains 100%.

Exploring the Landscape of Spatial Robustness

Engstrom et al. engstrom2017exploring investigate the vulnerability of neural network-based classifiers to rotations and translations. They focus on building robust spatial classifiers in computer vision applications using MNIST, CIFAR10, and ImageNet datasets. They perform analysis to compare different adversaries, including first-order, grid-based, and random, to develop attack meth. They consider data augmentation in training their model using standard training, no random rotations, no random cropping, and random rotations. They have improved the models' performance to 98%, 82%, 56% for MNIST, CIFAR10, and Imagesnet, respectively.

The Limitations of Deep Learning in Adversarial Settings

In this paper [9], authors creating a novel algorithm for creating adversarial examples using the knowledge of DNN architecture(white box attack) against deep learning. They build adversarial saliency maps between the input and the output of the DNN to explore the adversarial search space, and they used the forward derivatives in their methods. Their approach was developed with supervised NN. However, it can be used in unsupervised learning. This method was successfully misclassified a computer vision application with 97% using MNIST dataset with almost 4.02% as a perturbing average of input features per sample. They also explored defenses against adversarial samples in two methods: adversarial samples detection and improved DNN robustness by generating adversarial samples in the training time to augment the training set and be a regularizer method.

DeepFool: a simple and accurate method to fool deep neural networks

In [12], Moosavi et al. proposed a DeepFool algorithm that can compute adversarial examples and make the classifier more robust. They also indicate the concept

of generalizable, which means if an adversarial example created for a specific model can fool other models. They have compared their algorithm among different classifier robustness: binary and multicast classifiers. They also use their method in generating adversarial examples to augment and train a classifier with adversarial examples using the L2 norm. They tested the DeepFool algorithm on multiple classifiers, including LeNET, FC500, NIN, CaffNet, GoogLeNet with MNIST, and CIFAR-10 and ImageNet datasets.

ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models

Authors in [67], design black-box attacks by using zeroth-order optimization (ZOO) along with dimension reduction, hierarchical attack, and sampling techniques to attack black-box models in less time. They did their experiment using MNIST, CIFAR10, and ImageNet. The experiment results show that their ZOO attack is valid and can attack black-box DNNs. They compare the ZOO attack with Carlini Wanger's white-box attack. Also, they set up their experiment using a similar framework of Carlini Wanger. The ZOO attack decreases the probability of the original class and increases the probability of the target class, successfully attacking the model.

FAdeML: Understanding the Impact of Pre-Processing Noise Filtering on Adversarial Machine Learning

Khalid et al. [68] conducted a comprehensive analysis of pre-processing nose filter impact on adversarial attacks. They demonstrate that most of the state-of-the-art attacks can naturalize. They also proposed a new attack call FAdeML that able to defeat pre-processing noise filtering. They evaluate their attack on VGGNet DNN and German Traffic Sign Recognition Benchmarks (GTSRB) dataset.

Analysis of causative attacks against SVMs learning from data streams

In other words, Burkard et al. [50] examined a data stream of an active adversary that manipulate the training data on Support Vector Machine (SVM) while learning from a data stream. SVMs are vulnerable in which the attacker can modify the learner based and add new samples or the change labels in the training data. Moreover, evaluate the impact of this attack on the machine learning model performance by proposing a new metric and compare with the current metrics. They use as accuracy and empirical risk of an SVM model as metrics to evaluate if a model is learning well. They also used in their evaluation the predictive accuracy of the model using k-fold cross-validation. After analyzing the SVM models, they created an attack strategy under constraints and evaluated against learning system with anomaly detection defense. They used 2000 training samples for the input stream that contain 100 new samples, and they run the experiment using scikit-learn [50]. Strength: they define the general problem in attack machine learning because deceiving and defense machine learning models in a significant field. They built on past research, and their objective was stated clear. Weakness: They did not show the value of the metrics. They only used the graph for the result, which makes it unclear how they prove the claim. The evidence is not convincing. Also, they only discuss how these metrics can help in defense of the SVM model, but they did not prove it and included it in their experiment.

3.5.2 Related work in Adversarial Defences

Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems

Usama et al. [69] they proposed a black-box GAN-based adversarial attack to attack IDS. They also proposed training method based on GAN-based training. They used KDD99 and use feature extractions. They compared the effectiveness of the

GAN-based attack, and they attack various models, including DNN, support vector machine (SVM), random forest (RF), logistic regression (LR), decision trees (DT), k-nearest neighbor (KNN). Their results show that their proposed attack increase the false positives and make the classifier learn wrong boundaries. Also, their defense demonstrates definite improvement in the accuracy of IDS after training.

Adversarial Deep Learning for Robust Detection of Binary Encoded Malware

In [23], the authors presented four adversarial attack methods to generate an adversarial example of a binary malware file that preserves its functionality. They present a framework for training robust malware detection models by utilizing the saddle-point formulation that consists of the inner maximization and outer maximization problems. The inner maximization for generating effective adversarial examples that maximize the loss. Their framework success in building robustness model against adversarial examples during training by evaluating the evasion rates for the four adversarial attacks. They proposed a new metric for measuring the ration of adversarial malware modified during the training process. Their experiment result shows that rFGSM adversarial trained model has the lowest evasion rate and higher accuracy among all adversarial attacks. They used PE files as a dataset created from VirusShare. BCA and BGA attacks have relatively low evasion rates comparing to the FGSM attacks. However, BCA model has a high evasion rate among all adversaries attacks, which indicates less robustness than other models.

Towards Deep Learning Models Resistant to Adversarial Attacks

Mardy et al. identify various methods to attack and train NN models. They used these methods to train networks to significantly improve the resistance of different adversarial attacks. They used the saddle point min-max formulation to capture the

defense against the adversarial attack. They used MNIST and CIFAR10 in training NN against the PGD and first-order adversary. The MNIST model achieves 89% accuracy against a white-box attack with an iterative adversary. While the CIFAR10 model achieves an accuracy of 46%. [21]

CAT: Customized Adversarial Training for Improved Robustness

Cheng et al. [70] proposed an algorithm based on adversarial training called Customized Adversarial Training (CAT). CAT is based on min-max formulation adapts and customizes the perturbation level and can utilize any loss function. This method outperforms existing adversarial training methods using CIFAR-10. The built Wide-ResNet and achieve 73% robust against PGD attacks. At the same time, accuracy achieve 71% against the CW attack.

A Resilient Stream Learning Intrusion Detection (SLID) Mechanism for Real-time Analysis

[17] designed a resilient stream learning intrusion detection mechanism that is resiliency to adversarial attack and can update when facing new attacks. SLID analyze and measure real-time network traffic via stream analysis. They used to compute the communication statistics tumbling window. They designed that by generating a pool of classifiers, each one with a unique subset of features. The resiliency is reached by relying on voting before the model update to make sure the classifier is updated according to the pool of outcome. They selected three operation points for the tests: 1 as Low, 25 as Average, and 50 as High. To evaluate the SLID, they compared it with two other approaches the traditional machine learning (TML) and traditional stream learning (TSL). They used resiliency rate for evaluation where TML is 74.99 and 90.44 for the TSL while SLID reaches 92.53, 96.74, and 97.64 percent for the Low, Average, and High.

Feature Selection against Poisoning Attacks

Xiao et al. [71] provided a framework that investigates the robustness of popular feature selection methods like LASSO, ridge regression, and the elastic net. They evaluated the weakness of feature selection algorithms and using the poisoning attack. They considered in their experiment a malware detection application for detecting malware in PDF files based on present keywords rather than the content of data streams. They focused on their experiment setup on the feature that denoted the number of occurrences in the PDF file. They used 5993 recent malware samples from the contagion dataset and 5951 benign samples from the web strength: the analyses and the experiments have supported the author's significant findings and conclusions. They used classification error as a metric where poisoning up the training data causes the classification error to increase ten times as in LASSO; the classification error goes from %2 to%20. Weakness: the expected result did not occur when the ridge shows higher robustness under a poisoning attack. They found that poisoning the feature selection algorithm is much more effective than a random attack as in the presence of few poisoning samples, the attacker can arbitrarily control feature selection. A critical observation in this work is that the feature selection algorithm shows higher robustness under attack when it has more significant feature subsets. This leads to the importance of selecting large feature subsets against poisoning attacks and the relation in the ratio between the training set size, the dimensional feature set, and the impact of poisoning and evasion on the bias-variance decomposition of the mean squared error.

Detection of Adversarial Training Examples in Poisoning Attacks

Puadi et al. [20] proposed a defense and countermeasure mechanism based on outlier detection to mitigate the effect of poisoning attack against leaner classifiers. Adversarial examples generated by poisoning attacks are different from valid points, and they can be stopped by pre-filtering of the training dataset. Their evaluation of the real dataset shows the effectiveness of the proposed defense methodology, where the features numbers are high compared to training points. They show how more simplified constrained attacks are harder to detect. They performed an evaluation using a real dataset from CI repository:8 Spambase and MNIST, which are standard benchmarks in spam filtering and computer vision and include label flipping attack strategies. Their techniques are capable of stopping the optimal poisoning attacks. At the same time, it could not stop the label flipping attacks with their proposed defense because the attack points generated closer to the right points. This considers weakness in this work.

Poisoning Attacks and Defenses in Malware Detection Systems

Chen et al. [53] first, they examined how malware detection systems in the smartphone system, including DREBIN, MAMADRIOD, and DRIODAPIMINER can be misled and reduced accuracy by crafted malware samples and injected into training data. To solve this problem, [53], they designed a two-phase adversarial machine learning mechanism called KUAFUDET into mobile malware detection to reduce the poisoning attacks. The offline phase (Training Mode) includes selecting and extracting features from labeled applications dataset, and the Online Detection phase classifies the online Andriod application into multiple classes benign and malicious. This proposed detection is an automated detector that feedbacks the filtered suspicious false-negative into the training phase. Chen et al. evaluated the proposed

detection of more than 250,000 mobile application samples from Pwnzen Infotech Inc., including 10400 malicious samples that include threats such as phishing, spyware, and trojans proof that it can significantly reduce false negative and increase accuracy to %15 [53]. They measured the efficiency and scalability of the proposed detection and poisoning attack. The poisoning attack misleads the three detections DREBIN, MAMADRIOD, and DRIODAPIMINER, with a rate of 75.20%, 68.95%, 80.05%. The evaluation shows the highest accuracy in the proposed KUAFUDET 96.39% and STROMdroid 93.80% and Drebin 93.90%. The time also is not affected by the size of the data. The average detection time per application is 3 seconds. One of the limitations of the work is the number of features they selected 195 features that new malicious behaviors might disturb the features space and make the system less effective. Also, the limitation of the similarity-based approach can be inferred from the similarity threshold used. Limitation of decompilation technologies is another limitation—also the lack of a massive number of samples to scrutinize [53].

We reviewed work on machine learning and a deep learning-based intrusion detection system in Table 1. We also reviewed some recent research studies in adversarial attacks and defense in the intrusion detection fields, as shown in Table 2. From our observation, many research studies have widely studied and developed adversarial attack methods in computer vision and image recognition. However, adversarial attacks and defense in Intrusion Detection Systems are still in the early stages of research and have significant potential for further studies. Besides, Table 3 has recent research that uses adversarial training based on min-max formulation. We observed that most of the work in Table 3 is in image classification. Finally, we classified some researches in Table 4 based on the classification in Figure 5 in section 3.2.

Table 1: Overview of Machine learning-based and Deep learning-based Intrusion Detection System

Paper	Year	Algorithms	Datasets
Mahalanobis et al. [72]	2020	GAN	KDD99, SVHN, CIFAR10
Almiani et. al [73]	2020	RNN	NSD-KDD
Yang et al. [74]	2019	Auto-Encoder	NSD-KDD,UNSW-NB15
Yin et al. [36]	2019	CNN, LSTM, SAE	ISCX VPN-nonVPN traffic & ISCX 2012 IDS dataset
Shone et al. [75]	2018	DNN	NSL-KDD
Vijayanand et al. [76]	2018	SVM	Working environment traffic
Kwon et al. [77]	2017	DNN Survey	KDDCup 1999
Yin et al. [35]	2017	RNN	NSL-KDD
Wang et al. [37]	2017	CNN, LSTM	DARPA199, ISCX2012
Khalid et al. [78]	2016	Restricted Boltzmann Machine(RBM), Logistic Regression (LR)	KDDCup 1999
Javaid et al. [33]	2016	DNN	NSL-KDD
Corona et al. [79]	2013	Survey,Taxonomy, Solutions and open issue in IDS	

Table 2: Overview of adversarial attack and defense of Machine learning-based and Deep learning-based Intrusion Detection

Paper	Year	Algorithms	Datasets	Adversarial Attacks	Defense approach
Aboukhamis et al. [1]	2020	ANN	UNSW-NB15	FGSM, BGA, BCA	min-max formulation
Ayub et al. [80]	2020	MLP	CICIDS 2017,TRA-bID 2017	JSMA	NA
Usama et al. [69]	2019	GAN, DNN,SVM,RF,kNN	KDD99	Black-box attack	GAN-based training
Ibitoye et al. [62]	2019	FNN,SNN	UNSW BoT-IoT	FGSM, BIM, PGD	Feature Normalization
YePeng et al. [52]	2019	DNN,SVM,RF, Logistic Regression	NSL-KDD	FGSM,PGD,L-BFGS,SPSA	N/A
Wang et al. [3]	2018	Multilayer Perceptron(MLP)	NSL-KDD	FGSM,JSMA,Deep fool,CW	N/A
Homoliak et al. [7]	2018	Naive Bayes,Logistic Regression,Decision Tree,SVM	ASNM-NPBO	Evasion Attack	Obfuscations-aware classifier
Biggio et al. [64]	2013	SVM	HTTP-delivered attacks	Causative (Poisoning) Attack	N/A
Kloft et al. [63]	2010	Support vector data description (SVDD)	real HTTP traffic	Causative (Poisoning) Attack	N/A

Table 3: Overview of Adversarial Training and min-max optimization in Relevant Work. As shown in the table, only our papers in [1] [2] are utilizing min-max formulation in intrusion detection systems

Paper	Year	Dataset	Application	Algorithm	Attack
AbouKhamis et al [2]	2020	UNSW-NB15 , NSD-KDD	Intrusion Detection System	ANN,CNN,RNN	FGSM,BIM,PGD, CW,Deepfool
AbouKhamis et al [1]	2020	UNSW-NB15	Intrusion Detection System	ANN	FGSM, BGA, BCA
Shafahi et al. [81]	2020	CIFAR-10,ImageNet	Image Classification	Wide-ResNet	FGSM,PGD,Deepfool
Cheng et al. [70]	2020	CIFAR-10	Image Classification	Wide-ResNet	PGD, CW
Balaji et al. [82]	2019	CIFAR-10, CIFAR-100, Imagenet	Image Classification	Resnet, WideRenset	PGD
Wang et al. [83]	2019	MNIST and CIFAR-10	Image Classification	MLP, All-CNN, LeNet, LeNetV2, VGG16, GoogLeNet	PGD
Zhang et al. [84]	2019	MNIST and CIFAR-10	Image Classification	CNN	PGD, CW
Yan et al. [57]	2018	MNIST, CIFAR-10, ImageNet	Image Classification	MLP, LeNet, ConvNet, ResNet, etc	Deepfool, FGS
Al-Dujaili et al. [23]	2018	Malware Binary file	Malware Detection	ANN	FGSM, PGD
Shaham et al. [85]	2018	MNIST and CIFAR-10	Image Classification	ANN	FGSM, BIM
Raghunathan et al. [86]	2018	MNIST	Image Classification	NN	PGD
Tramer et al. [87]	2018	MNIST	Image Classification	CNN	FGSM,PGD
Ding et al. [88]	2018	MNIST, CIFAR10	Image Classification	DDN-Rony,TRADES, CNN	Adaptive Norm PGD
Wong et al. [22]	2017	MNIST	Image Classification	CNN	FGSM,PGD
Mardy et al. [21]	2017	MNIST and CIFAR-10	Image Classification	CNN	FGSM,PGD,CW

Table 4: Classification of relevant works in Adversarial Attacks based on Figure 5

Paper Information		Machine Learning Info			Adversarial Surface	Adversarial Capabilities	Adversarial Goals	Adversarial Samples Generation	
Paper	Year	Application	Algorithms	Data sets				White or Black-Box	Attack Methods
AbouKhamis et al. [1]	2020	IDS	ANN	UNSW-NB15	Evasion	Data injection	Targeted	WhiteBox	FGSM, GBA, CBA
AbouKhamis et al. [2]	2020	IDS	ANN,CNN,RNN	NSD-KDD,UNSW-NB15	Evasion	Data injection	Targeted	WhiteBox	FGSM, PGD,BIM,CW, Deepfool
Ibitoye et al. [62]	2019	IDS	DNN (FNN,SNN)	BoT-IoT	Evasion	Data injection	Targeted	WhiteBox	FGSM, BIM, PGD
Usama et al. [69]	2019	IDS	GAN	KDD99	Evasion	Data injection	Targeted	BlackBox	Blackbox
Peng et al. [52]	2019	IDS	DNN	NSL-KDD	Evasion	Data modification	Targeted	WhiteBox	FGSM,PGD,BFGS,SPSA
Wang et al. [3]	2018	IDS	DNN	NSL-KDD	Evasion	Data modification	Targeted	White Box	FGSM,JSMA,Deep fool,CW
Homoliak et al. [7]	2018	IDS	NB,LR,ST, SVM	ASNM-NPBO	Evasion	Data modification	Targeted	WhiteBox	Non-payload-based Obfuscation
Biggio et al. [64]	2013	IDS	SVM	HTTP Traffic Attack	Poison	Data Injection	Targeted	WhiteBox	Gradient Based
Kloft et al. [63]	2010	IDS	Support vector data description (SVDD)	Real HTTP Traffic	Poison	Data modification	Targeted	White Box	Input Manipulation
Khalid et al. [68]	2019	Image Classification	VGGNet	GTSRB	Evasion	data modification	Targeted	white box	FGSM,BIM,L-BFGS
Chen et al. [67]	2017	Image Classification	DNN	MNIST, CIFAR, ImageNet	Poison	Data modification	Targeted	BlackBox	Carlini,Wanger (CW)
Burkard et al. [50]	2017	Binary Classifier	SVM	Dataset stream	Poison	Data modification	Targeted	WhiteBox	sample generation
Mossavi et al. [12]	2016	Image Classification	DNN	MNIST, CIFAR, ILSVRC	Poison	Data modification	Targeted	WhiteBox	Deepfool
Li et al. [89]	2016	Collaborative Filtering Systems	DNN	MovieLens Dataset	Poison	Data modification	Targeted	WhiteBox	Gradient based
Mei et al. [27]	2015	Multiple Applications	SVM,Logistic Reg.,Linear Regression	Wine quality, OLS, Spambase	Poison	Data modification	Targeted	WhiteBox	Label Manipulation
Biggio et al. [90]	2011	SpamBayes spam Filter model	SVM	Breast-cancer,Diabetes,Heart dataset	Poison	Data modification	Targeted	WhiteBox	Label Manipulation

Chapter 4

Problem Statement and Methodology

This chapter covers our research problem statement that carried out the purpose, scope, and thesis direction. Also, we present our methodology, including the IDS framework architecture and algorithm.

4.1 Problem Statement

DNNs are increasingly used in security applications such as network intrusion detection systems [6] [3] [7] [8]. The high volume of network traffic increases the number and type of attacks and the challenges to label and classify the network traffic data. The fact that IDS is adversarial in nature, researchers make security aspects of DNN increasingly important and a critical design goal, especially against adversarial samples. Adversarial samples have intentionally designed to target a model to degrade performance and result in incorrect decisions with high confidence. This wrong output can be done by adding a calculated perturbation to the input. The linear nature of DNN is the main reason for its vulnerability against adversarial perturbation. While deep learning-based IDS trained models aim to be very effective in classifying benign and malign inputs and reducing false alarms, adversarial samples often expose blind

spots in the input. The fact that adversarial samples can simply be generated using different algorithms can be used to perform attacks against different models that lead to misclassifying the input. Several works have study adversarial attacks for DNN in image classification [21] and text recognition. However, adversarial attacks and defense techniques in intrusion detection models are still in the early stages of research and have significant potential for further studies. The protection and robustness against adversarial attacks is a growing challenge [62] [14] [8] and should always be addressed by DNN researchers and designers [17] [18]. This served as a potential and motivation to explore deep learning-based IDS and focus on popular types of deep neural networks algorithms, including ANN, CNN, and RNN, on different benchmark datasets and study the effectiveness of different well-known and state-of-the-art adversarial attacks against DNN and their robustness.

Researches on adversarial machine learning concentrate on two main points: how to generate capable adversarial samples that can attack and deceive a model with small perturbation and how to train and defend a model to be robust against adversarial samples. In this dissertation, our primary goal is to explore and study various deep neural network architectures on two benchmark IDS datasets: UNSW-NB15 and NSD-KDD in the adversarial environment from both the attack and defense approach. We utilize an IDS adversarial training framework presented in Abou-khamis et al. [1] based on a min-max formulation that can improve the robustness of the DNN based IDS against effective adversarial samples. We evaluate the IDS framework present in Abou-khamis et al. [1] on the NSL-KDD and UNSW-NB15.

4.2 Research Methodology

We use the following methodology to carry out the research. We describe the details and justify our approach by our experimental results. In this research, we have

conducted three main experiments divided them into three Phases, as this allows us to do a comprehensive analysis between the results. The experiments that we conduct are presented in this research to study and analyze the effect of the min-max approach on adversarial attacks during the DNN inferences on two benchmark datasets: UNSW-NB15 and NSD-KDD. In chapters 5, 6, and 7, we present our experimental results to validate our selected approach.

We start with a brief of our experiment methodology. We then describe the two datasets and how we preprocess them. The next section is about our DNN models Architecture and learning setup for each set of experiments. Then we illustrate our experiment process.

1. We train the DNNs models in two different phases: Phase I and Phase II. We first split UNSW-NB15 and NSD-KDD into training, testing, and validation sets.

2. We then preprocess the datasets: i) convert the non-numerical features, ii) normalize the datasets, apply feature selection using two techniques: PCA and Recursive Feature Elimination (RFE). The experimental results of applying two feature selection techniques, determine which feature technique to use with the entire experiments in Phase II and Phase III.

3. As a baseline, we train six models, two of each ANN, CNN, and RNN on the two adversarial-free datasets: UNSW-NB15 and NSD-KDD. We then used the baseline ANN, CNN, and RNN models to generate a collection of adversarial samples from the five adversarial attack methods: FGSM, BIM, PGD, CW, and DeepFool using inner-maximization problem.

4. We then utilize Algorithm I based on [1] to retrain the ANN, CNN, and RNN models, where each model architectures with two datasets.

Algorithm I Experimental Process

```
Input: Dataset, Inner-maximizer(), evasion_method
Output: adv_model: Robust Adversarial Trained Model, x*_train, x*_test: adversarial samples, Accuracy
Epoch = 10
Batch_size=32
Extract Important Features(x_train,x_test)
x_train, x_test = train_test_split (Dataset)
model <- Construct model (Epoch, Batch_size)        // using different algo e.g, ANN, CNN and RNN
If evasion_method  != "Attack_free" Then            //FGSM, PGD, BIM, CW or DeepFool
   x*_train, x*_test<- Inner-maximizer(x_train, x_test, evasion_method)  //generate adversarial samples
 End if
adv_model<-Adversarial_Training(model, x*_train)   //training using adversarial samples
Accuracy <- Evaluate(adv_model, x*_test)    //evaluate the accuracy of adversarial trained model
```

5. In Phase II, we have conducted various sets of experiments to evaluate the effectiveness of the adversarial samples of five evasion adversarial attack methods generated using inner-maximizer.

6. We retrained using the min-max approach to evaluate the robustness of the deep learning-based IDS.

7. Phase III experiments conducted to evaluate the robustness of adversarial models that are retrained using Algorithm I against one specific attack using a min-max approach against different adversarial attacks. We called this set of experiment Matrix experiments.

We denote the evasion attack methods as FGSM, CW, BIM, PGD, and DeepFool. All experiments were performed on two traffic flow datasets and evaluated based on prediction accuracy.

The adversarial training process based on min-max approaches is formalized in ***Algorithm I*** [1]. The Algorithm describes the IDS framework solution based on min-max formulation 3.

We implement ***Algorithm I*** to take input and craft adversarial samples using

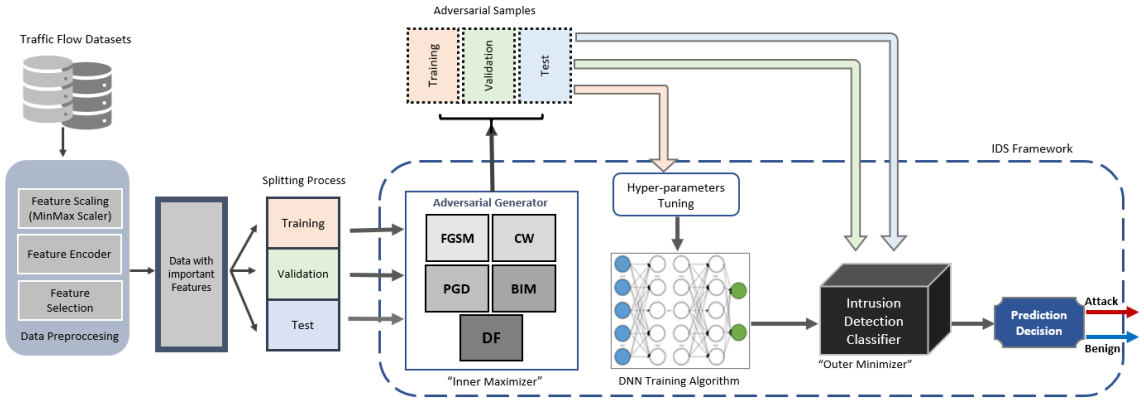


Figure 6: IDS Framework includes Inner Maximizer that generates adversarial samples and maximizes the loss and Outer Minimizer that minimizes the loss presented in [1]

the "inner maximization" based on maximization formula 2. It generates adversarial versions that maximize the loss of $\mathcal{T}$. Then, we harden the six models against the crafted adversaries using the "outer minimization" based on minimization formula 1 by augmenting them in the training process. The deep learning-based IDS framework solution is illustrated in Figure 6. This Figure illustrates how the dataset splits into three sets: training, validation, and testing. It also shows how the entire dataset passes the inner-maximizer to generate the strongest adversarial samples and then augment them in the learning phase to minimize the loss. We evaluate in this research the impact of min-max approach as follows:

- *F*irst, we evaluate the quality of the adversarial samples found on UNSW-NB15 and NSK-KDD datasets generated by maximization formula 2 against the six baseline models.

- Second, evaluate the adversarial IDS robustness against adversarial samples generated from the same attack method were each deep learning architecture in a different experiment.

- Third, evaluate the adversarial IDS robustness against adversarial samples generated from different adversarial methods.

- All our evaluation performed using test data: 25,195 and 50,7863 traffic flow on NSD-KDD and UNSW-NB15, respectively.

We evaluated the studied min-max methods on the UNSW and NSL-KDD dataset of a packet flow, where the task is to classify packet flow into one of the two classes: benign or attack. In this work, we consider some assumptions about our attack threat model. The attacks are evasion attack, where an attacker has access to our IDS model during prediction time that leads to misclassify the model decision and target the positive "attack" sample to be classified as a negative "benign" sample taking into considerations that a complete knowledge of the targeted models is known to perform a white-box attack with multi iterations. The adversarial threat model, including goal, capabilities, and knowledge of all the adversarial attacks are the same across all experiments.

We use a sufficient number of samples in both datasets for training, validation and test, and we split the training dataset into training and validation set to evaluate and assess the performance of the models during training time with 80% and 20%. In all experiments, we tested the adversarial IDS using new adversarial samples not used in the training time. Using different samples in training and testing guarantee that the model evaluated using not seen samples. The details of ANN, CNN, and RNN experiments are described in the sections below.

All IDS models training process and evaluation work the same way across all our models. We will specify the training configuration in each experiment in terms of the selected optimizer, loss function, parameters, and metrics in Chapter 5.

Chapter 5

Experimental Approach

Part of the experiments that we conducted is presented in this chapter to study and analyze the effect feature selection during the DNN inferences on two IDS benchmark datasets: UNSW-NB15 and NSD-KDD.

We start with the platform that we used for our experiments. We then describe the two datasets and how we preprocess them. The next section is about our DNN models Architecture and learning setup for each set of experiments. Then we illustrate our experiment process.

In Phase II, we have conducted various sets of experiments to evaluate the adversarial samples' effectiveness of five evasion adversarial attack methods generated using maximization equation 2. Also, to evaluate the robustness of the deep learning-based IDS framework architectures: ANN, CNN, and RNN are retrained using the min-max approach 3.

We use a sufficient number of samples in both datasets, and we split the training dataset into training and validation set to evaluate the performance of the models with 80% and 20%.

Generally, in our experiment, our model's training and evaluation work the same way across all our models. We specify in each experiment the training configuration in terms of the selected optimizer, loss function, parameters, and metrics.

5.1 Development Platform and Tools

All our models were implemented using TensorFlow and Keras. The experiments were performed on Virtual machine with GPU acceleration with 32 GB memory and three Intel core processors 2.294 GHz. For generating adversarial samples, we use the open-source IBM Robustness Toolbox (ART) framework [91].

5.2 IDS Datasets

Since the performance of deep learning models to some extent, depends on the dataset, model accuracy varies between different datasets. Therefore, it is significant to study the dataset carefully as part of the machine learning approach. Generally, the dataset used for IDSs is flow or packets. Different benchmark datasets are used for IDS by many researchers, including KDD99, NSL-KDD, and UNSW-NB1515. This leads to new research results to be compared to previous results. In our research, we use two datasets: UNSW-NB15 and NSL-KDD.

5.2.1 UNSW-NB15 Dataset

UNSW-NB15 dataset [5] is a raw network flow created by the IXIA perfecStorm tool in a lab that generates normal activities and attack behaviors.

The greater the number of output nodes the higher complexity you will add to your model. This means that given a fixed amount of data, a greater number of output nodes will lead to poorer results. I would use a ABCD vs. others strategy.

TCPdump is the used tool to capture the raw traffic. UNSW-NB15 contains nine different attacks, which mean multiple labels (or classes) includes: DoS, worms, Backdoors, and Fuzzers. Training with multiple classes and more output nodes increase the model complexity. Therefore, in our current work, we use the mentioned two classes

(attack and benign) only and leave the multi-class for further experiments. Each row in the dataset has 49 features, including class labels, and it has more than two million records. We list the details of features in Table 22 based on the description in [5]. These features include packet-based and flow-based features. The classification of the features is based on the three following classes: basic, content, and time. The basic features include connection information like protocols, ports, addresses. The content features have information about packet payload. Time features are about the start time and end time. Each row in the UNSW-NB15 dataset is labeled as 0 for normal, and 1 for attack records. Each attack record is also labeled with a particular class of attack. Also, the features in "UNSW-NB15" have various data types include integer, float, binary, timestamp, and nominal. Nominal data contains non-numerical (categorical) values, which DNN cannot process them directly. Therefore, we exercise data preparation before feeding our models in next section. In our experiments, we used 1,17478 records for training and 57,863 for testing, including 56,000 benign samples, and 119,341 attack samples. Having multiple classes in the classification is a long process. Therefore, in our experiments, we use binary classification with two classes: benign and attack labels only and leave the multi-class for further experiments.

Table 22 in the Appendix provides a summary of dataset features.

5.2.2 NSL-KDD Dataset

NSL-KDD is a new version of KDD 99 dataset that solve various problems of the KDD 99 dataset [92] [93]. Researches confirm that NSL-KDD still has inherited some of the problems discussed by McHugh and may not be a perfect representative of existing real networks because of the lack of public data sets for network-based IDSs. However, NSL-KDD is still considered as an efficient benchmark flow dataset that helps researchers [93] [94] [95] compare different intrusion detection methods. NSL-KDD has large number of training and test records. This advantage makes it

affordable to run the experiments on the complete set without the need to select a small portion randomly. NSL-KDD has no duplication in the entire dataset. It has 21 attack type in the training dataset and 37 attacks in the testing dataset grouped into DoS, Probe, U2R, and R2L. In our experiments, we use binary classification with two classes: benign and attack labels only. We conduct our experiments and train all deep learning models on a large dataset divided between benign and attack flow. Our training set consists of approximately 100,778 flow and 25,195 to the testing set, including 58,630 attack samples and 67,343 benign samples. Table 23 in the Appendix provides a summary of the attack types.

5.3 Data Preprocessing

Before we feed the model with the data, we need to preprocess the datasets. We perform different methods for preprocessing on both selected datasets. Both NSL-KDD and UNSWB datasets have different scales and have some significant outliers. To improve the predictive performance trained model, prevent slow training time, we remove outliers by normalize all feature values in the dataset into the range of [0,1] using MinMax Scaler. We choose to go with MinMax Scaler because it does not change the features embedded in the original data. We use "LabelEncocder" function to convert the categorical (or non-numeric) features into numeric. We encode three columns in the NSK-KDD dataset: protocol, service, and flag. While in the UNSW dataset, we encode protocol, service, and state columns.

Before building the CNN and RNN models, we reshaped the train and test input into three dimensions (3D) using reshape() function. CNN and RNN are generally used in computer vision applications with images as input where images are represented in a 2D or 3D array. In our research, we reshape the input into a matrix of size (100778 x 5 x 1) for training set and size of (25195 x 5 x 1) testing set for NSL-KDD.

We also reshape the size of (117478 x 5 x 1) for training set and size of (57863 x 5 x 1) for the testing set for the UNSW-NB15 dataset.

5.3.1 Feature Selection

Both selected datasets UNSW and NSL-KDD are large and have a large number of features. Preprocessing a big dataset is a challenging process an leads to cause a bottleneck during processing. Often, many of the features are insignificant and redundant and may lead to inefficiency in the prediction process because they have less contribution in prediction time comparing with other essential features. Also, they may consider as noise and lead to increase the training time and overfitting. Therefore, the feature selection process, as shown in Figure 9 is significant to select essential features and to enhance the performance of the model in terms of prediction accuracy and simplicity of the underlying process and faster prediction process. Fewer features in training time reduce the mode complexity.

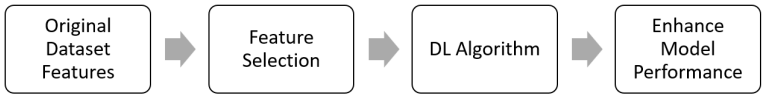


Figure 7: Feature Selection Process.

There are several algorithms for feature selection. Feature selection algorithms could be based on statistical measures that rank the feature and apply the score to determine if the feature will be kept or removed. Another type of feature selection is called the Wrapper method that assigns scores by evaluating a group of features based on model accuracy and which groups of features contribute the most of the prediction process. A search process selects the set of features. Recursive Feature Elimination (RFE) algorithm is an example of this method. While learning which features contribute best to the model accuracy during the model building process is called the Embedded method. Regularization is an example of this method. In

the following section, we focus on investigating experimentally the best feature selection methods that can be applied to achieve high accuracy and better performance of deep learning-based IDSs. We use two different selection techniques: Recursive Feature Elimination (RFE) and Principal Component Analysis. Experiment details and results of feature selections are discussed in Chapter 6

5.4 Architecture Characteristics & Learning Setup

In the section, we explore the selected parameters for building the three DNN architectures: ANN, CNN, and RNN, considering selecting the optimum hyper-parameters that have a significant impact on model performance. A model output in classification time depends on the how the selected algorithms work and how changing the hyper-parameter influences the model prediction. We use in building the models architectures different parameters, including neuron numbers in layers, hidden layers, learning rates, and dropout. Learning Rate should be very small to achieve satisfactory results. It manages the number of weights alteration. On the other hand, a high number of neurons in hidden layers leads to robust neural networks but will require more time to train the model. The number of hidden layers usually is selected based on the dataset size and dimensions.

5.4.1 ANN-IDS Architecture

In ANN experiments, we implement an Artificial Neural Network Architecture that achieves performance comparable with other studies in this area. We use the same ANN architecture for UNSW-NB15 and NSL-KDD datasets. We use the ADAM optimizer, and we slice the data into batches with a size of 32 and repeatedly over the

entire dataset over ten epochs to train the model. The accuracy obtained is 96.64% for the UNSW-NB15 dataset and 96.07% for NSL-KDD. Our network parameters and architecture listed in Table 5.

Table 5: Training Parameters and Architecture for IDS Models in ANN Experiments

Parameter	Value
No. of hidden layers	3
Layer 1	128 neurons
Layer 2	96 neurons
Layer 3	64 neurons
Dropout	0.25
Optimizer	ADAM
Activation function	ReLU and Softmax
Learning rate	0.01
Epoch	10
Batch Size	32

We run the experiments in this set using the above ANN architecture on two different dataset UNSW-NB15 and NSL-KDD. for two-class classification. First, we train the model using the original dataset to built an adversarial-free ANN model. Then, we attack the adversarial-free ANN models using five different attacks: FGSM, CW, BIM, PGD, and Deepfool. We then retrain the model using Algorithm I.

5.4.2 CNN-IDS Architecture

In this set of experiments, we choose to build the CNN model because the Convolutional layer has been proven that significantly reduce the training time. We use the same CNN architecture for both UNSW and NSL-KDD datasets and the same

hyperparameters. Our CNN network architecture has a combination of three types of layers: Convolution, pooling, and fully connected layers. Our CNN architecture has three Conv1D that extract features from the packet flow. It also has two MaxPooling1D that apply dimensionality reduction to reduce the inputs' size and decrease computation time. We use four fully connected layers with the ReLU activation function and one output layer with a softmax activation layer to classify the input into one of the categories: benign or attack. We also use a flatten layer to flatten the input shapes and Dropout layers to prevent overfitting in the model during training time.

Our layers flow includes Conv1D $->$ MaxPooling1D $->$ Conv1D $->$ MaxPooling1D $->$ Conv1D $->$4 Fully connected layers$->$ Output layer.

We then use in our CNN model ADAM optimizer, and we slice the data into batches with a size of 32 and repeatedly over the entire dataset over ten epochs to train the model. The accuracy obtained is 96.19% for the UNSW-NB15 dataset and 95.69% for NSL-KDD. Our CNN network parameters listed in Table 6.

We run the experiments in this set using the above CNN architecture on two different dataset UNSW-NB15 15 and NSL-KDD. We train the model using the original dataset to build an adversarial-free CNN model. Then, we attack the adversarial-free CNN models using five different attacks: FGSM, CW, BIM, PGD, and Deepfool. We then retrain the model using Algorithm I.

5.4.3 RNN-IDS Architecture

In this section, we used in our experimental setup RNN with LSTM for long-term learning on two different dataset UNSW-NB15 15 and NSL-KDD. We use same LSTM architecture for both UNSW and NSL-KDD datasets and same hyperparameters. Our RNN network architecture has a combination of three types of layer. We use two LSTM layers with sigmoid activation function and one fully connected layers

Table 6: Training Parameters and Architecture for IDS Models in CNN Experiments

Parameter	Value
Convolution layer	3 layers, L1:28, L2:42, L3:128 neurons
Max Pooling	2 layers
Dropout	0.25
Fully Connected	4 layers, L1:512, L2:256, L3:128, L4: 64 neurons
Output layer	Softmax
Optimizer	ADAM
Activation function	ReLU and Softmax
Learning rate	0.01
Epoch	10
Batch Size	32

with ReLU activation function and one output layer with softmax activation layer to classify input into one of the categories: benign or attack.

ADAM optimizer is used, and we slice the data into batches with a size of 32 and repeatedly over the entire dataset over ten epochs to train the model. The accuracy obtained is 95.71% for the UNSW-NB15 dataset and 95.47% for NSL-KDD. Our RNN network parameters listed in Table 7.

We train the model using original data sets to build an adversarial-free RNN model. Then, we attack the adversarial-free RNN model using five different attacks: FGSM, CW, BIM, PGD, and Deepfool and then retrain the model using Algorithm I. Our results for all experiments on two data sets summarized in Table 17.

Table 7: Training Parameters and Architecture for IDS Models in RNN Experiments

Parameter	Value
LSTM layer	2 layers with Sigmoid
Fully Connected	1 layer with ReLU and 128 neurons
Dropout	0.5
Output layer	Softmax
Optimizer	ADAM
Activation function	ReLU , Softmax and Sigmoid
Learning rate	0.01
Epoch	10
Batch Size	32

5.5 Evaluation Metrics

Before diving into the results and evaluation, we want to present the measurement metrics we use to evaluate the results in all experimental phases. Specifically, to help compare various model robustness in the adversarial environment and the effectiveness of the adversarial attacks on the models. Different metrics can be used to evaluate the performance, including model accuracy (ACC), Precision Rate (PR), Recall Rate (RR), F-Score (FS), and the area under the receiver operating characteristics curve (AUC). In our research, we refer to **attack traffic flow** to **Positive**. While, the **benign traffic flow** to **Negative**. Our models predict a binary outcome, whether correct prediction (True Positive or True Negative) or wrong prediction (False Positive or False Negative).

Confusion Matrix shows the four outputs for performance measurement for the classifier: True Positive, True Negative, False Positive, False Negative, as shown in Figure 8. The definitions are listed in Table 8.

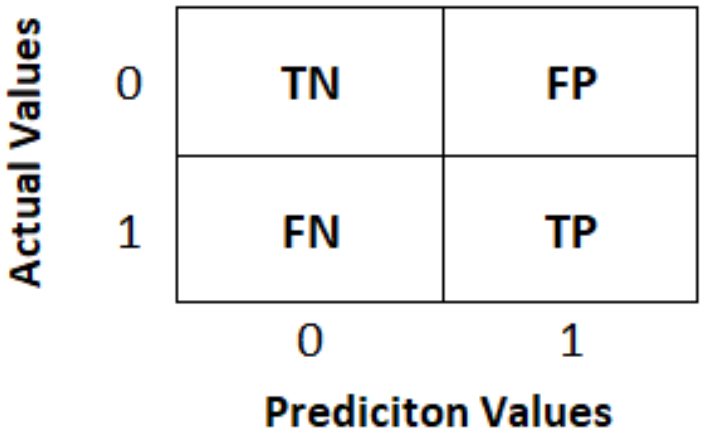


Figure 8: Confusion Matrix

Table 8: Definitions of Metrics Terms [3] [4]

Term	Definitions
True Positive (TP)	Number of attack samples that is correctly classified as attack
False Positive (FP)	Number of benign sample misclassified as attack
True Negative (TN)	Number of benign sample that is correctly classified as benign
False Negative (FN)	Number of attack samples misclassified as benign

The following metrics formulas are based on True and False positive and True and False-negative [96] [4].

- *Prediction Accuracy (AC)*: Total number of correctly classified samples from benign and attack samples among all number of all samples.

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

- *Specificity* : It is a number of true negatives among samples classified as negative. Also, we refer to it as True Negative Rate (TNR):

$$Specificity = TNR = \frac{TN}{TN + FP}$$

- *Sensitivity (or Recall)*: It is also called True Positive Rate (TPR) or detection rate. It is a total number of True Positives (TP) among all actual positive samples:

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

- *Precision*: The ratio of real positive samples over the total predicted positive samples.

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

- *False Positive Ratio (FPR)*: It is the inverted specificity and also called a false alarm rate.

$$FPR = \frac{FP}{FP + TN} = 1 - Specificity$$

- *False Negative Ratio (FNR) or Evasion Rate (ER)* : The Ratio of positive samples that have mispredicted as negative samples. Also, we refer to is as an error rate. It should be as low as possible.

$$FNR = EV = \frac{FN}{FN + TP} = 1 - TPR$$

- F-Score: It measures a more realistic model performance with both Recall and precision simultaneously. It is used when the dataset has uneven class distribution—value range between 0 and 1. A value near 1 represents a balance between

Recall and precision. While value near 0 appears when having low Recall and precision

$$F - Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

- Receiver Operating Characteristics Area Under Curve (ROC AUC): This plot is the recall and precision values contrary to each other.

We use ROC to evaluate our binary models' overall performance among several algorithms, ANN, CNN, RNN. It determines the best algorithms where the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis. The ROC curve plots the false detection or alarm versus detection probability [97]. The model here is giving a probability between the range 0 and 1. To make these white and black predictions, we need to select the decision boundary between this range. The Area Under the Curve (AUC) is a number that reflects the model performance. The best model will have a higher AUC toward one [98]. AUC is a secure method to interpret the model performance [99] [98]. If the ROC curve shows small values on the x-axis in the graph, this reflects that we have low false positive and high true negative. If the ROC curve shows larger values in the y-axis, we have a high true positive and low false negative. The perfect model curve should travel from the bottom left to the top right of the plot, and the AUC should be near 1, while AUC near 0 indicates a poor model. We can measure accuracy, Recall, Precision, Specificity, F-score, and AUC-ROC curve using the confusion matrix. In Chapter 6, we used confusion matrix to evaluate and compare the baseline models performance.

Chapter 6

Phase I: Performance of adversarial-free Deep learning-based IDS

Our objective in Phase I is to build highly accurate deep learning models on UNSW-NB15 and NSD-KDD for each architecture: ANN, CNN, and RNN as baseline models (benchmarks) to comprehensively compare the performance of deep learning-based IDSs in the adversarial environment in Phase II. We build six baseline IDS models, and we refer to them as following: ANN-UNSW, ANN-KDD, CNN-UNSW, CNN-KDD, RNN-UNSW, and RNN-KDD. The same experimental setup was repeated using two feature selection techniques: RFE and PCA, to compare the results between the two feature selection techniques in terms of prediction accuracy. Phase I experiment results of applying two feature selection techniques, determine which feature technique to move forward with the entire experiments in phases II and Phase III.

6.1 Evaluation of Feature Selection Experiments

In this section, We focus on investigating experimentally the best feature selection methods that can be applied to achieve high accuracy and better performance of deep learning-based IDSs. We use two different selection techniques, Recursive Feature

Elimination (RFE) and Principal Component Analysis.

Table 9: The Top seven features using RFE feature selection from UNSW-NB15 and NSD-KDD datasets

Selected Features			
ID	UNSW-NB15	ID	NSD-KDD
2	protocol	4	service
8	wrong-fragment	9	dbytes
22	is-guest	10	rate
29	same-srv-rate	13	sload
30	diff-srv-rate	14	dload
35	dst-host-diff-srv-rate	22	stcpb
39	dst-host-srv-serror-rate	23	dtcpb

Table 10: adversarial-free ANN, CNN and RNN Accuracy Percentage using RFE

-	Predictive Accuracy		
Dataset	ANN	CNN	RNN
UNSW-NB15	90.21	89.02	87.33
NSD-KDD	89.59	89.54	86.97

6.1.1 Performance using RFE

First, we use RFE with the Logistic Regression to assess the usefulness of our features. Therefore, we limit the selection to the essential features in the selected datasets. We train the ANN, CNN, and RNN using RFE feature selection for UNSW-NB15 and NSD-KDD. Based on the rank, we selected the top seven features, as shown in Table 9.

The experiment was executed multiple times for each DNN algorithm: ANN, CNN, RNN with different batches, and epoch values to obtain the best accuracy. The

accuracy of ANN, CNN, and RNN models on UNSW-NB15 and NSL KDD using RFE with the top seven features achieve 88.77% as an average. All experiments results using RFE for all models on the two datasets are listed in Table 10.

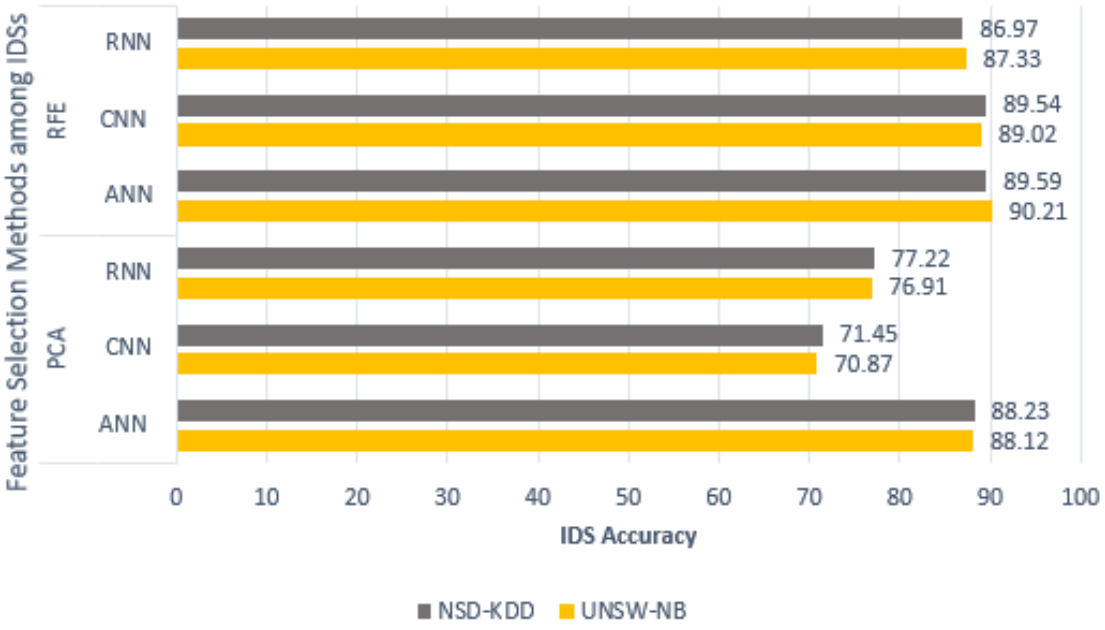


Figure 9: Comparison Between PCA and RFE Feature Selection Accuracy among ANN, CNN and RNN IDS

6.1.2 Performance using PCA

Similarly, with the same experimental setup and parameters, we train the ANN, CNN, and RNN using PCA on UNSW-NB15 and NSD-KDD. We select a different number of principal components to reduce the high dimension of our dataset while making sure not to lose necessary information during reduction and keep variation. With PCA, we select some of the dataset features based on original feature correlations. We repeated the experiments three times with different number of principal components 15, 10, and 5, as shown in Table 11. When we selected five principal components, the accuracy of the CNN model achieves almost 71% for UNSW-NB15 and KDD, while

the RNN model achieves almost 77% for UNSW and NSL KDD. However, ANN accuracy achieves almost 88% for UNSW-NB15 and NSD-KDD, as shown in Table 11.

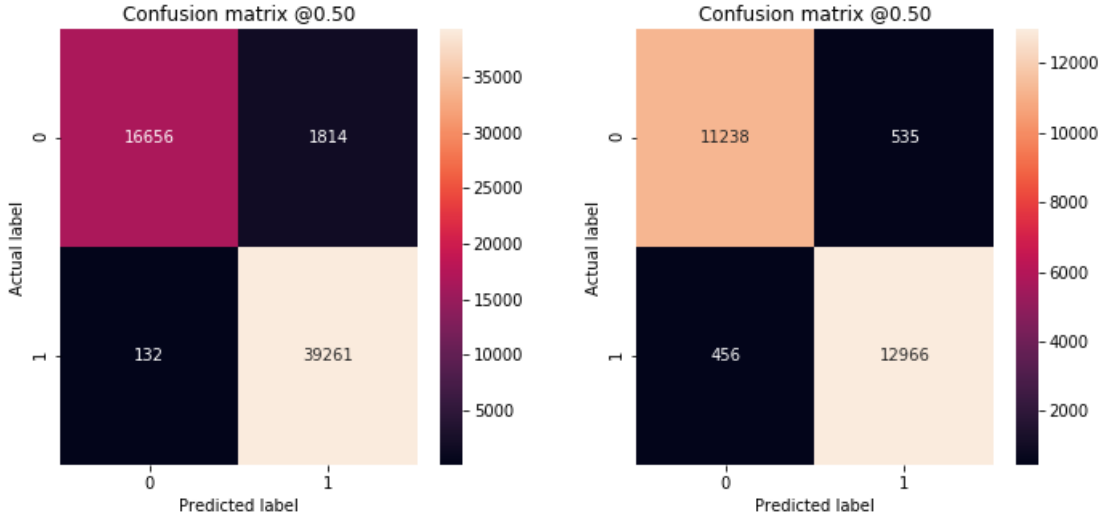
Table 11: ANN, CNN and RNN Accuracy percentage using PCA with different number of principle components

-	15			10			5		
Dataset	ANN	CNN	RNN	ANN	CNN	RNN	ANN	CNN	RNN
UNSW-NB15	75.19	63.53	61.34	75.33	66.72	66.55	88.23	71.45	77.22
NSD-KDD	79.86	71.36	75.00	80.45	70.97	76.57	88.12	70.87	76.91

Table 12: The Top 5 Selected features from UNSW-NB15 and NSD-KDD Datasets

Selected Features			
ID	UNSW-NB15	ID	NSD-KDD
2	protocol	4	service
8	wrong-fragment	9	dbytes
22	is-guest	10	rate
29	same-srv-rate	13	sload
30	diff-srv-rate	14	dload

We can observe from Table 10 and 11 that accuracy with RFE for all models architectures ANN, CNN, and RNN was higher than accuracy achieved using PCA feature selection. The comparison between PCA and RFE is illustrated in Figure 9. However, we still did not achieve the accuracy that we expect. After various tries and observations, we improve our model performance by selecting five attributes listed in Table 12 after assessing the model accuracy and avoiding model overfitting. Our deep learning models trained using the features in Table 12 achieve noticeably better accuracy than models trained using PCA and RFE with seven feature selection techniques. At this stage of understanding, we believe this might be possible because



(a) UNSW-NB15 (b) NSD-KDD

Figure 10: Confusion Matrix for ANN on NSD-KDD & UNSW-NB15

we use regularization techniques for prepossessing the two selected datasets.

6.2 Baseline Deep learning-based IDS Results in Adversarial-free Environment

In this section, our objective is to find the most accurate deep learning model and powerful binary classifiers on the two selected datasets: UNSW-NB15 and NSD-KDD for ANN, CNN, and RNN. We train each model on the same 80% of the dataset and conduct the validation on the remaining 20%. The experimental results of the three deep learning-based IDS architectures trained on two datasets: UNSW-NB15 and NSD-KDD are illustrated in Table 13. We use a confusion matrix to describe our experimental performance of the binary classification models. We also use the ROC curve as shown in Figures 13 and 14 to exhibit the performance of the binary classification models in an adversarial free environment where the higher value of

the AUC indicates how accurate the classifier prediction. We observe that the deep learning-based IDS models perform well on prediction time with unseen data. The performance of each architecture is described in the following sections.

6.2.1 Performance of ANN IDS in Adversarial-free Environment

We report ANN models' performance and discuss their capability to learn benign and attack traffic flow on UNSW-NB15 ad NSD-KDD data sets. Figure 13 and 14 show the ROC curves of two ANN-based IDS models on UNSW-NB15 and NSD-KDD datasets in an adversarial-free environment. The ANN detection models achieve satisfactory AUC scores over 99% for UNSW-NB15 and equal to 98% for NSD-KDD. High AUC value toward 1 represents the best model, as indicates in [98].

Figure 10 illustrates the total number of prediction sample results for ANN in the testing dataset. TP=39261 samples correctly predicted as an attack, TN= 16656 samples correctly predicted as benign, FP=1614 samples wrongly predicted as an attack, FN= 132 samples wrongly predicted as benign for UNSW-NB15. Similarly for NSD-KDD, TP=12,966, TN= 11,238, FP=535 and FN= 456. We noticed from the results that the TPR is very high, and FNR is significantly low on both datasets.

Table 13 shows the performance metrics results of ANN models in an adversarial-free environment to assist in understanding and comparing the model performance. The result values are rounded to the nearest integer.

- *Accuracy* that represents the total perdition of the model is 97% and 96% for UNSW-NB15 and NSD-KDD, respectively.

- *Precision* that represents the total number of real attack samples over the total attack predicted samples is 97% and 96% for UNSW-NB15 and NSD-KDD, respectively.

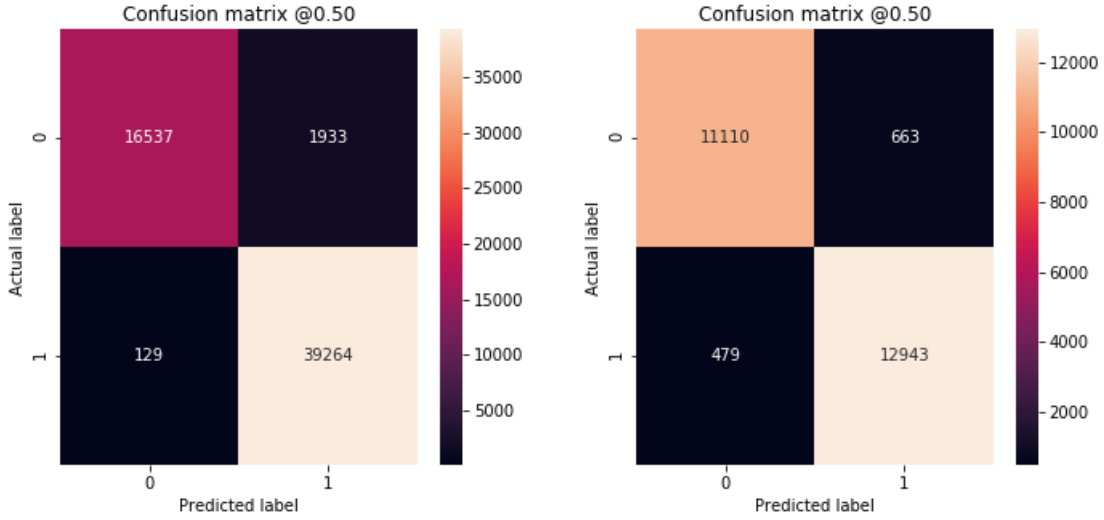
- *Recall (TPR)* that represents the total number of real attacks over the actual number of attacks is 97% and 96% for UNSW-NB15 and NSD-KDD, respectively.
- *F-score* value is 97% and 96% for UNSW-NB15 and NSD-KDD, respectively. The values in both datasets are near one which reflects the balance between recall and precision.

Table 13: Evaluation Metric Results including prediction Accuracy, precision, recall of ANN models

Dataset	Accuracy	Precision	Recall	F-score
UNSW-NB15	0.97	0.97	0.97	0.97
NSD-KDD	0.96	0.96	0.96	0.96

6.2.2 Performance of CNN IDS in Adversarial-free Environment

In this section, we will report the performance of the CNN models and discuss their capability to learn benign and attack traffic flow on UNSW-NB15 ad NSD-KDD data sets. Figure 13 and 14 show the ROC curves of two CNN based IDS models on UNSW-NB15 and NSD-KDD datasets in a clean environment. The CNN detection models achieve satisfactory AUC scores over 99% for UNSW-NB15 and equal to 98% for NSD-KDD. High AUC value toward one represents the best model [98]. Figure 11 illustrates the total number of prediction sample results for ANN in the testing dataset. TP=39264 samples correctly predicted as an attack, TN= 16537 samples correctly predicted as benign, FP=1933 samples wrongly predicted as an attack, FN= 129 samples wrongly predicted as benign for UNSW-NB15. Similarly for NSD-KDD, TP=12,943, TN= 11,110 , FP=663 and FN= 479. We noticed from the results that the TPR is very high, and FNR (error rate) is significantly low on both datasets.



(a) UNSW-NB15 (b) NSD-KDD

Figure 11: Confusion Matrix for CNN on NSD-KDD & UNSW-NB15

Table 14 shows the performance metrics results of ANN models in an adversarial-free environment to assist in understanding and comparing the model performance. The result values are rounded to the nearest integer.

- *Accuracy* that represents the total perdition of the model is 96% and 96% for UNSW-NB15 and NSD-KDD, respectively.

- *Precision* that represents the total number of real attack samples over the total attack predicted samples is 95% and 96% for UNSW-NB15 and NSD-KDD, respectively.

- *Recall (TPR)* that represents the total number of real attacks over the actual number of attacks is 100% and 96% for UNSW-NB15 and NSD-KDD, respectively.

- *F-score* value is 97% and 96% for UNSW-NB15 and NSD-KDD, respectively. The values in both datasets are near one which reflects balance between recall and precision.

Table 14: Evaluation Metric Results including prediction Accuracy, precision, recall and F-score of CNN models in adversarial-free environment

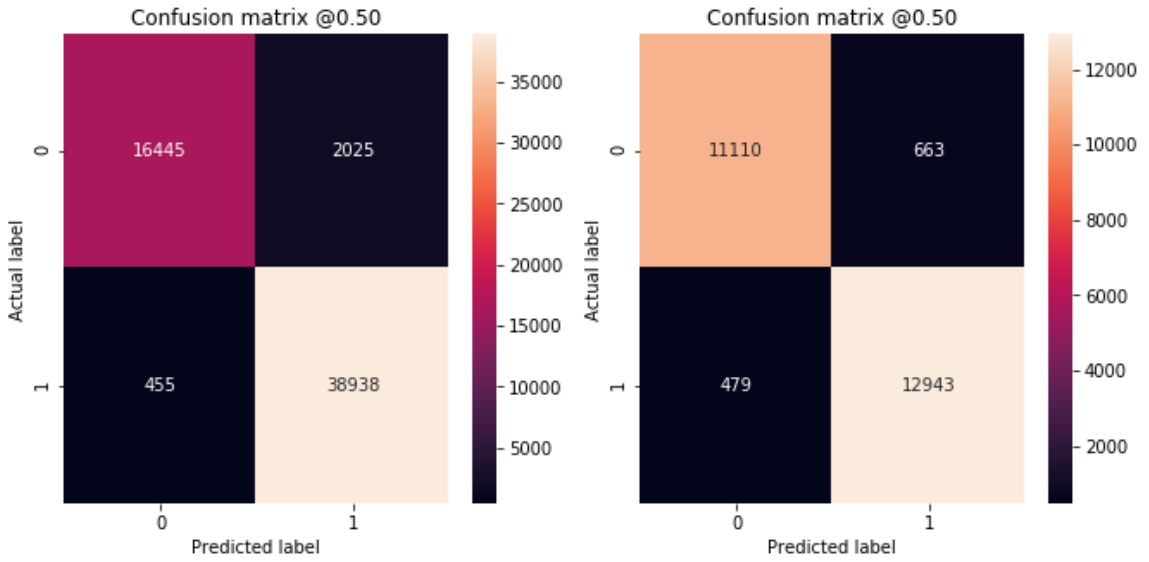
Dataset	Accuracy	Precision	Recall	F-score
UNSW-NB15	0.96	0.97	0.96	0.96
NSD-KDD	0.96	0.96	0.96	0.96

6.2.3 Performance of RNN IDS in Adversarial-free Environment

In this section, we will report the performance of the RNN models and discuss their capability to learn benign and attack traffic flow on UNSW-NB15 ad NSD-KDD data sets. Figure 13 and 14 show the ROC curves of two RNN based IDS models UNSW-NB15 and NSD-KDD datasets in the adversarial-free environment. We also noticed that the RNN detection models achieve satisfactory AUC scores over 99% for UNSW-NB15 and equal to 98% for NSD-KDD. High AUC value toward one represents the best model [98]. Figure 12 illustrates the total number of prediction sample results for ANN in the testing dataset. TP=39264 samples correctly predicted as an attack, TN= 16537 samples correctly predicted as benign, FP=1933 samples wrongly predicted as an attack, FN= 129 samples wrongly predicted as benign for UNSW-NB15. Similarly for NSD-KDD, TP=12,943, TN= 11,110, FP=663 and FN= 479. We noticed from the results, that the TPR is very high and FNR (error rate) is significantly low on both datasets. Table 15 shows the performance metrics results of ANN models in an adversarial-free environment to assist in understanding and comparing the model performance. The result values are rounded to the nearest integer.

Accuracy that represents the total perdition of the model is 96% and 95% for UNSW-NB15 and NSD-KDD, respectively.

Precision that represents the total number of real attack samples over the total attack



(a) UNSW-NB15 (b) NSD-KDD

Figure 12: Confusion Matrix for RNN on NSD-KDD & UNSW-NB15

predicted samples is 96% and 95% for UNSW-NB15 and NSD-KDD, respectively. *Recall (TPR)* that represents the total number of real attacks over the actual number of attacks is 94% and 95% for UNSW-NB15 and NSD-KDD, respectively. *F-score* value is 95% and 95% for UNSW-NB15 and NSD-KDD, respectively. The values in both datasets are near one which reflects the balance between the recall and precision.

Table 15: Evaluation Metric Results of RNN models in adversarial-free environment

Dataset	Accuracy	Precision	Recall	F-score
UNSW-NB15	0.96	0.96	0.94	0.95
NSD-KDD	0.95	0.95	0.95	0.95

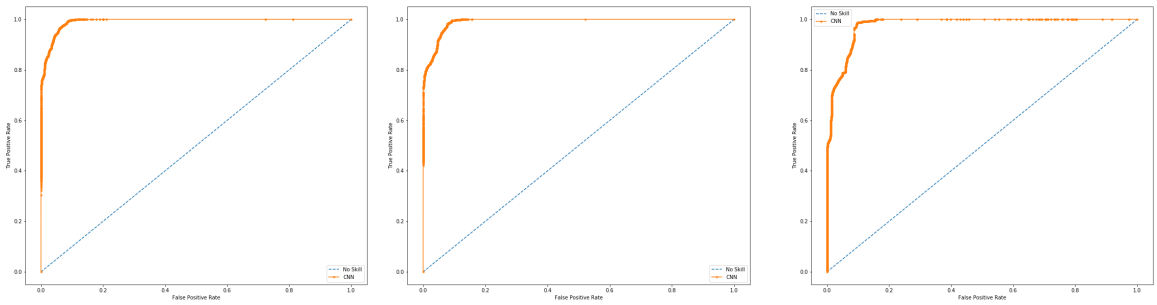
As shown in Table 16, the baseline prediction accuracy for all six baseline IDS models needed for the evaluation of the performance in an adversarial environment.

In this phase, we build highly accurate deep learning models on UNSW-NB15 and

Table 16: Prediction Accuracy (%) for ANN, CNN, and RNN IDSs in adversarial-free environment

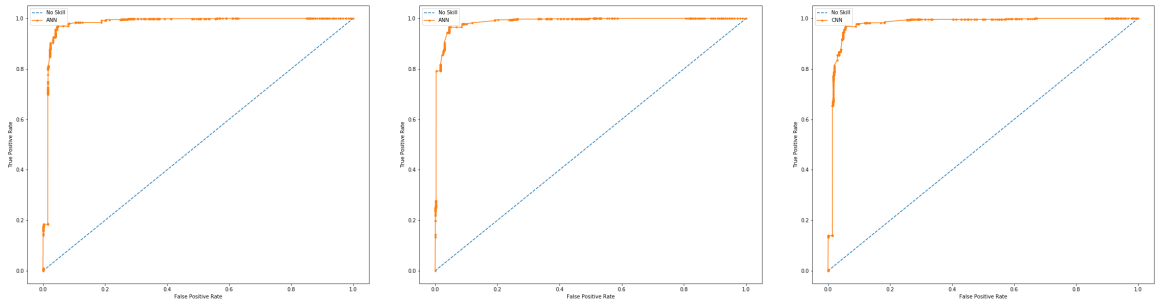
Dataset	ANN	CNN	RNN
UNSW-NB15	95.71	96.19	96.64
NSD-KDD	95.47	95.69	96.07

NSD-KDD for each architecture: ANN, CNN, and RNN as baseline models (benchmarks) to comprehensively compare the performance of deep learning-based IDSs in the adversarial environment in Phase II and focus on investigating experimentally the best feature selection methods that can be applied to achieve high accuracy. We use two different feature selection techniques: Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA). After evaluating Phase, I experiment on different deep learning algorithms: ANN, CNN, and RNN. We found that RFE is the right choice for both UNSW-NB15 and NSD-KDD. After building a highly accurate baselines model, we will dive into adversarial attacks and defense experiments in the next two chapters.



(a) ANN (b) CNN (c) RNN

Figure 13: ROC curve for ANN, CNN and RNN baseline IDSs in adversarial-free environment on UNSW-NB15 Datasets



(a) ANN (b) CNN (c) RNN

Figure 14: ROC curve for ANN, CNN and RNN baseline IDSs in adversarial-free environment on NSD-KDD Datasets

Chapter 7

Phase II: Evaluation of the Deep Learning-based IDS Framework with a Single Adversarial Attack

This Chapter reports and evaluates the Deep Learning-based IDS framework with a single adversarial attack. We divided Phase II into two parts: Phase II-1 and Phase II-2. Our experiment results illustrate the effectiveness of the five adversarial attacks: FGSM, CW, PGD, BIM, DeepFool generated using maximization problem against a deep learning-based IDS: ANN, CNN, and RNN on test data: UNSW and NSL-KDD. Also, results illustrate the robustness of the IDS models trained using min-max formulation.

- In Phase II-1, we start finding adversarial samples from the five state-of-the-art adversarial attack methods with high perturbation that maximize the loss using inner-maximizer.
- Evaluate the quality of the adversarial samples found on UNSW-NB15 and NSK-KDD datasets generated by inner-maximizer against the six baseline IDS models.

- In Phase II-2, we retrain the models with the same experimental setup in Phase I using Algorithm I with a single adversarial method to minimize the maximal loss. With three architectures, two datasets, and five adversarial attack methods, we build 30 adversarially trained IDS models in Phase II.2.

- Evaluate the adversarially trained IDS model robustness against adversarial samples generated from the same adversarial method.

The fact that we use different datasets, deep learning algorithms, and architectures and different adversarial attack methods, we report the results and findings of the experiments from different angles [100]:

- Adversarial attack methods vs. the min-max defense technique based on model architectures

- Model architectures vs. adversarial attack methods or defense based on the dataset

- Dataset vs. Dataset based on adversarial attack methods and defense Our experimental results show that min-max during training with a single type of adversaries significantly increases the accuracy and the robustness of the adversarially trained models. With this behavior, we indicate adversarial training successfully solving the min-max optimization problem.

The precise goal of our adversarial attacks is to misclassify the attack packet flow as a normal traffic flow and to increase the false negative (FN). In our research, we intend to prove that the adversarial training based min-max formulation is considered a reliable defense technique against different adversarial attacks in DNN. The experimental results also show how the inner-maximizer technique has substantial impacts on the adversarial attacks and how it produces persuasive adversarial samples that maximize the loss.

Threat model assumptions: we consider some assumptions about our attack threat model. The attacks are evasion attack, where an attacker has access to our IDS model during prediction time that leads to misclassify the model decision and target the positive "attack" sample to be classified as a negative "benign" sample taking into considerations that a complete knowledge of the targeted models is known to perform a white-box attack with multi iterations. The adversarial threat model, including goal, capabilities, and knowledge of all the adversarial attacks are the same across all experiments. More details for result evaluations are discussed in the following sections.

7.1 Phase II.1: Adversarial Attacks Experiments Using Inner-Maximizer

We study and evaluate the effectiveness of the adversarial samples of five methods generated by inner-maximizer formula 2 against the six adversarial-free DNN models which they were built in Phase I. As a reminder, we refer to the six baseline IDSs to ANN-UNSW, ANN-KDD, CNN-UNSW, CNN-KDD, RNN-UNSW, and RNN-KDD.

In Phase II-1 experiments, we compare ANN, CNN, and RNN IDSs on both datasets: UNSW-NB15 and NSD-KDD. It is not surprising that the prediction accuracy among all baseline models have fallen sharply relative to adversarial-free environments. This is expected behavior because the models are not trained by any adversarial samples, and they are only trained by clean datasets.

7.1.1 Performance of ANN IDS under Adversarial Attacks on UNSW-NB15 and NSD-KDD

Figure 15 compares the effect of different adversarial attack methods against ANN IDS. We can demonstrate that the ANN IDS accuracy was degraded by the adversarial samples generated from the two datasets using the maximization problem. We achieve accuracy of 96.64% and 96.07% with no attack for UNSW-NB15 and NSD-KDD, respectively.

The prediction accuracy of ANN-UNSW IDS is reduced from 96.64% to 53% using FGSM adversarial samples. Similarly, as shown in Figure 15 with CW, BIM, PGD, and Deepfool adversarial sample, the accuracy of ANN-UNSW IDS degraded to 71.62%, 43.27%, 43.29%, and 37.87% respectively.

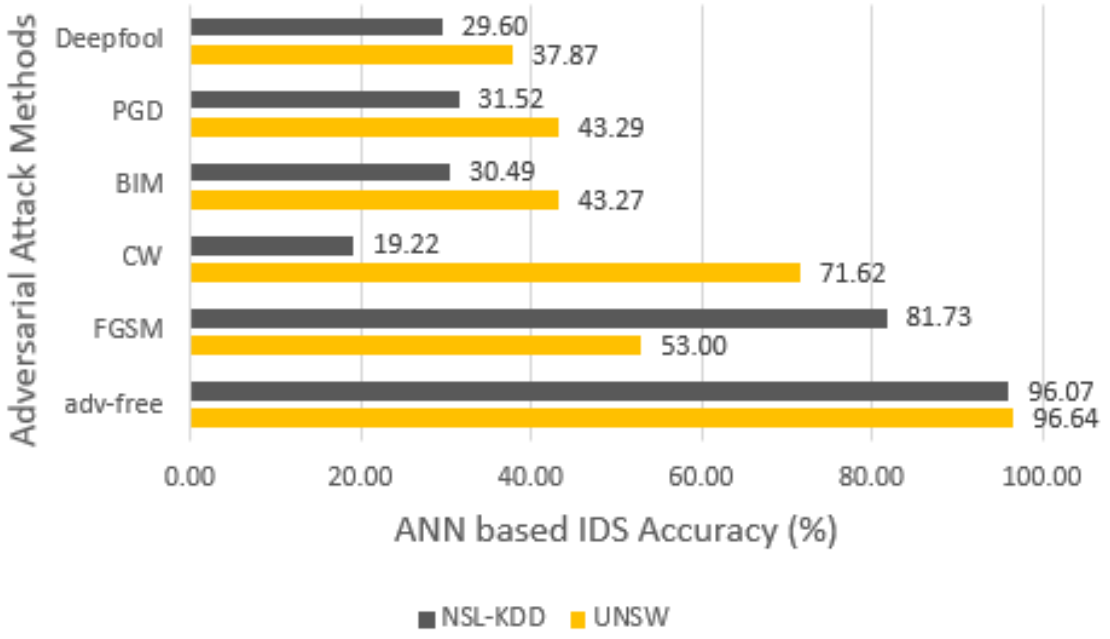


Figure 15: Effect of Adversarial samples generated by inner-maximizer on ANN trained by adversarial-free NSD-KDD and UNSW-NB15 Datasets

For NSD-KDD, the prediction accuracy is reduced from 96.07% to only 81.73% using FGSM adversarial samples. At the same time, accuracy decreased significantly

to 19.22%, 30.49%, 31.52%, and 29% using CW, BIM, PGD, and Deepfool adversarial samples, respectively.

We can conclude that the performance of Free-attack ANN-UNSW IDS was more resilient to the five adversarial attacks compared to ANN-KDD IDS. We also find that the FGSM attack has the least influence on both ANN IDS models. BIM attacks on both ANN models show an unsurprising result. The robustness of ANN was less against BIM compared to FGSM. In contrast, the Deepfool attack method has the highest effectiveness on both ANN models. PGD and BIM attacks have a similar effect on both ANN models. CW samples have a different impact on the two datasets. We can see that CW has a significant powerful impact on the NSD-KDD accuracy comparing to UNSW-NB15.

7.1.2 Performance of CNN IDS under Adversarial Attacks on UNSW-NB15 and NSD-KDD

We can demonstrate that the CNN IDS accuracy on 16 was also degraded by the adversarial samples generated from the two datasets using the five adversarial attacks generated by the inner-maximization. We achieve accuracy of 96.19% and 95.69% with no attack for UNSW-NB15 and NSD-KDD, respectively. As shown in Figure 16 with CW, BIM, PGD, and Deepfool adversarial sample, the CNN-UNSW accuracy degraded to 66.20%, 44.52%, 46.45%, and 66.94% respectively. The prediction accuracy of CNN-UNSW IDS is reduced from 96.14% to 71.31% using FGSM adversarial samples. We also noticed that CW and Deepfool strength are equal, same as BIM and PGD. FGSM attack has the least effect on CNN-UNSW IDS.

For NSD-KDD, the prediction accuracy is reduced significantly from 95.69 % to 32.48% using FGSM adversarial samples. Also, we noticed a significant decrease in about 70-80% of the initial accuracy with no attack. Accuracy decreased to 23.37%,

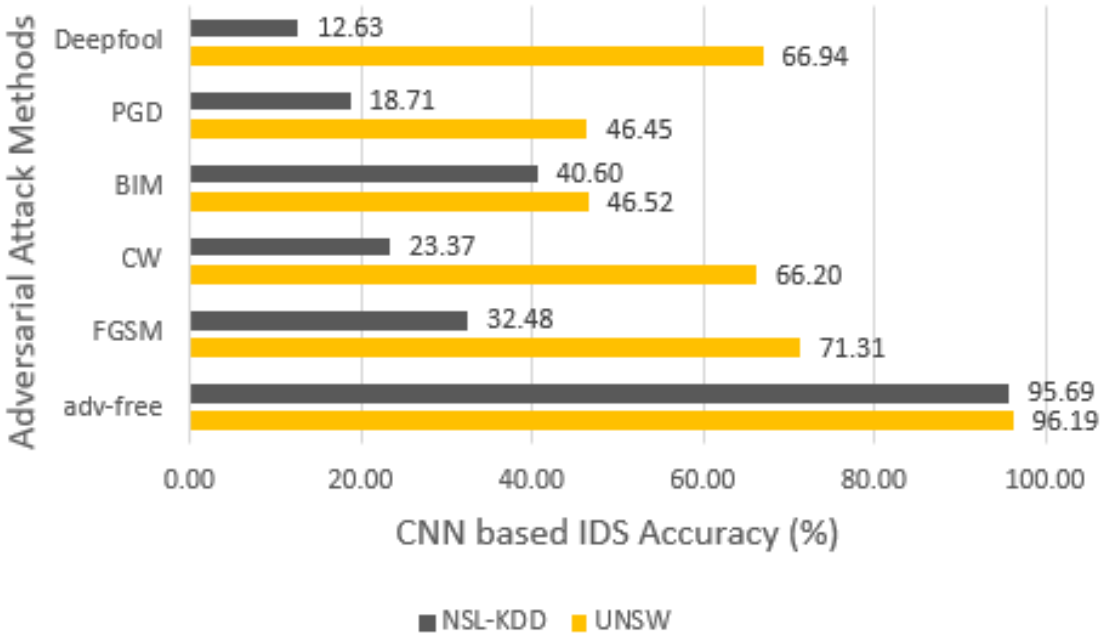


Figure 16: Effect of Adversarial samples generated by inner-maximizer on CNN trained by adversarial-free NSD-KDD and UNSW-NB15 Datasets

40.60%, 18.71%, and 12.63% using CW, BIM, PGD, and Deepfool adversarial samples, respectively, as shown in Figure 16.

Adversarial-free CNN IDS trained on UNSW-NB15 was more resilient to the five adversarial attacks than trained on NSD-KDD. It can be clearly seen that the five evasion attacks' effectiveness was more robust compared to the effect on the CNN model train by adversarial-free UNSW-NB15 dataset. We observe that BIM and PGD attacks are more destructive and decrease the accuracy by almost 50%. While FGSM, CW, and Deepfool attacks have less effectiveness on CNN-UNSW IDS. In CNN-KDD, CW, PGD, and Deepfool have the same and higher effectiveness comparing to BIM and FGSM attacks.

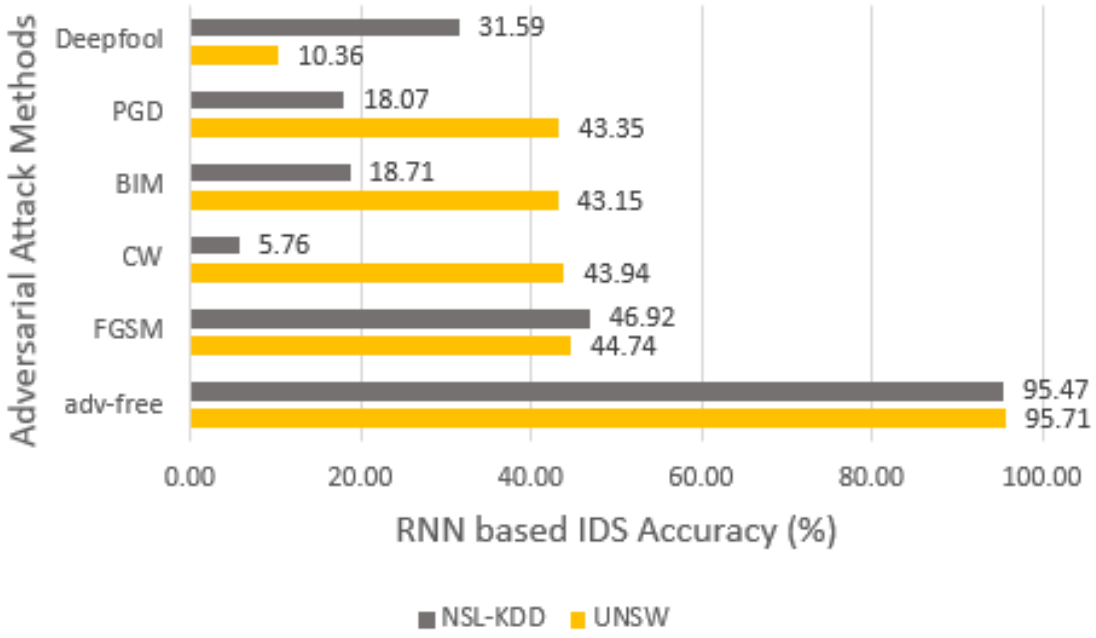


Figure 17: Effect of Adversarial samples generated by inner-maximizer on RNN trained by adversarial-free NSD-KDD and UNSW-NB15 Datasets

7.1.3 Performance of RNN IDS under Adversarial Attacks on UNSW-NB15 and NSD-KDD

Consistently with the result of the UNSW-NB15 and NSD-KDD on CNN and ANN experiments, it can be seen from Figure 17 that the RNN models IDS accuracy on the UNSW-NB15 and NSD-KDD was also degraded by the adversarial samples generated from the two datasets using the five adversarial attack methods. We achieve an accuracy of 95.71% and 95.47% with no attack for UNSW-NB15 and NSD-KDD, respectively.

As shown in Figure 17, the prediction accuracy of RNN-UNSW IDS is reduced by almost 50% percent from the original accuracy. We also noticed that all of FGSM, CW, BIM, and PGD consider having the same effect on RNN-UNSW IDS. Deepfool attack has the most considerable influence on the RNN-UNSW IDS.

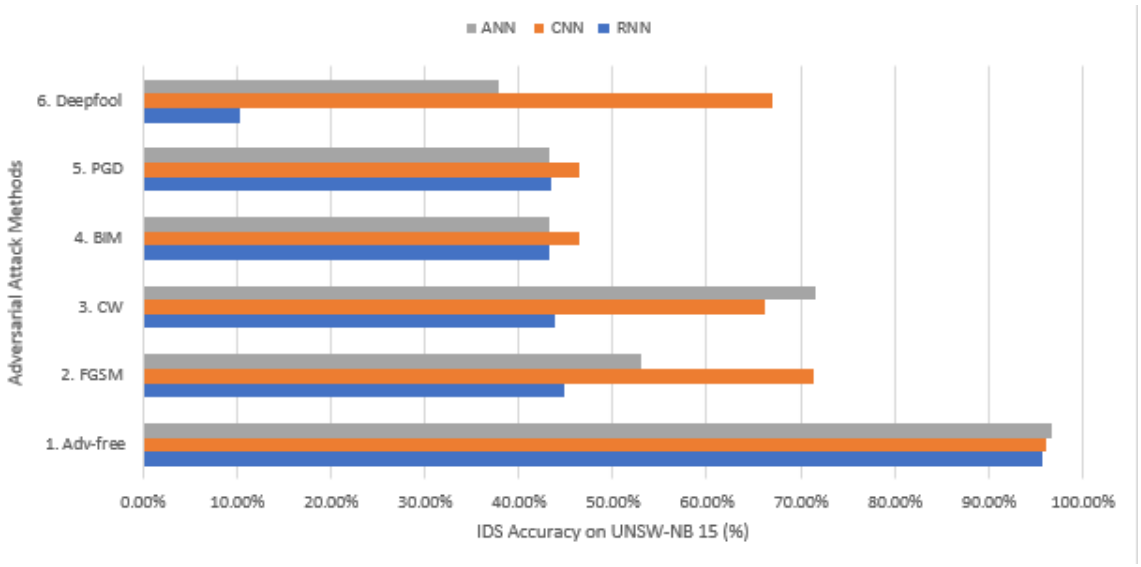


Figure 18: Effect of Adversarial samples generated by inner-maximizer on ANN, CNN and RNN based IDS trained by Adversarial-free UNSW-NB15

The performance of RNN-KDD IDS in the adversarial environment decreases sharply, and it is not robust in the adversarial environment because FGSM, CW, BIM, PGD, and Deepfool easily reduce the accuracy 46.92%, 5.76%, 18.71%, 18.07%, and 31.59% respectively. We notice the RNN IDS models have an inferior performance on Deepfool samples and CW samples, and both RNN IDS models do not exhibit any kind of robustness against CW and Deepfool attacks.

As a conclusion for RNN-UNSW IDS, FGSM, CW, BIM, and PGD attacks have all the same effectiveness. They decrease the initial accuracy significantly by the half. However, the Deepfool attack decreases the accuracy outstandingly by almost 85%. CW attack against RNN-KDD outstanding other attacks, where they reduce the accuracy sharply by 90%.

7.1.4 Performance Comparison of ANN, CNN and RNN IDSs Robustness in Adversarial Environment

The performance of ANN, CNN and RNN based IDS in adversarial environment is clear as shown in Figure 18 and19.

If we look closer to each IDS model in Figure 18 and evaluate its performance against all adversarial samples, we find BIM and PGD attacks have almost the same effectiveness against All IDSs. They decrease the accuracy by 50%. We observe RNN-UNSW IDS have the least resilience against the CW attack, while ANN-UNSW is more resilient to the CW attack. RNN-UNSW IDS accuracy decreases outstandingly by Deepfool attack. While CNN-UNSW IDS was highly robust against Deepfool attack. FGSM attack has almost the same effectiveness against ANN-UNSW and RNN-UNSW IDS while the least effective on CNN-UNSW IDS.

While we observe in Figure 19 that IDSs are less resilient to the adversarial attacks than IDSs trained on UNSW-NB15, as shown in Figure 18. With close analysis of each IDS, we find ANN-KDD IDS has similar resilience to BIM, PGD, and Deepfool. CNN-KDD IDS is the least resilient to the Deepfool attack. While RNN-KDD IDS are the most vulnerable to CW samples. RNN-KDD IDS is more vulnerable to most of the adversarial samples, while ANN-KDD IDS is more resilient to all attacks, particularly FGSM attacks comparing to other IDSs.

Discussion

Through the comprehensive comparison regarding the adversarial attacks against baseline IDSs, we can draw the following findings:

- FGSM Attack For ANN, CNN, and RNN models on NSL-KDD and UNSW datasets has less impact on almost all baseline models because the FGSM attack's purpose is to be fast, not optimal attack [101] [48].

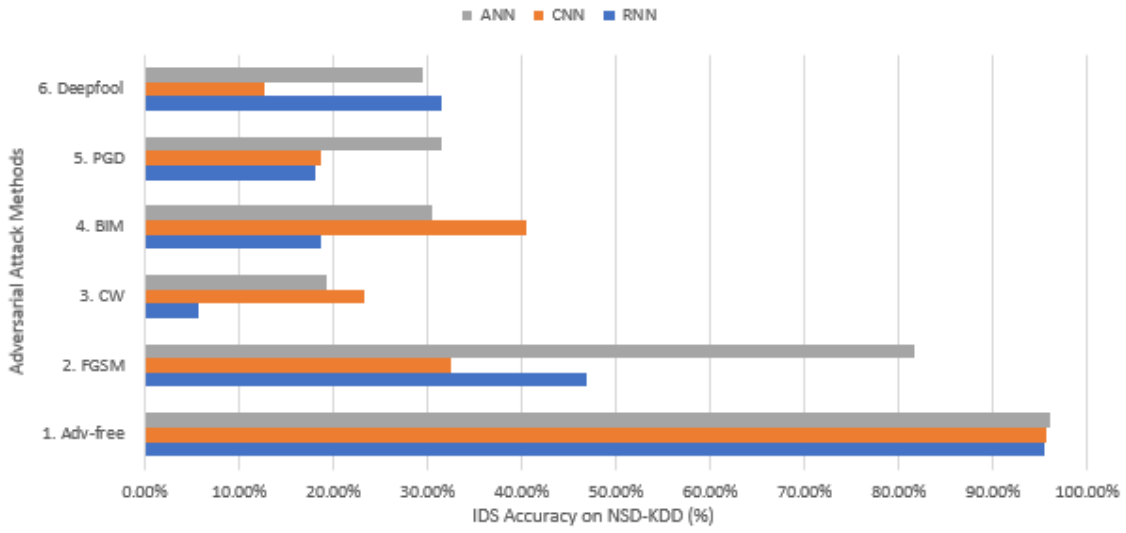


Figure 19: Effect of Adversarial samples generated byinner-maximizer on ANN, CNN and RNN based IDStrained by Adversarial-free NSD-KDD

- We observe that BIM and PGD attacks have similar effectiveness in UNSW-NB1515 and NSD-KDD experiments. However, BIM and PGD effectiveness against CNN-KDD are varying.

- CW attack considers potent attacks against baseline IDSs in the NSD-KDD experiments. However, CW attacks one of the least effective attacks against baseline IDSs in the UNSW-NB15 experiments.

- As we observe that the DeepFool attack shows the superiority over all other attacks expects for CNN-UNSW. Deepfool attack was considered a very strong attack [12] that can generate small perturbations to evade deep learning models.

- FGSM, CW, BIM, and PGD attacks have relatively same effectiveness in the UNSW-NB15 experiment.

- To summarise the UNSW-NB15 experiment, we observe that all baselines for ANN, CNN, and RNN IDS trained by adversarial-free samples are more resilient to the five adversarial attacks comparing to NSD-KDD experiments.

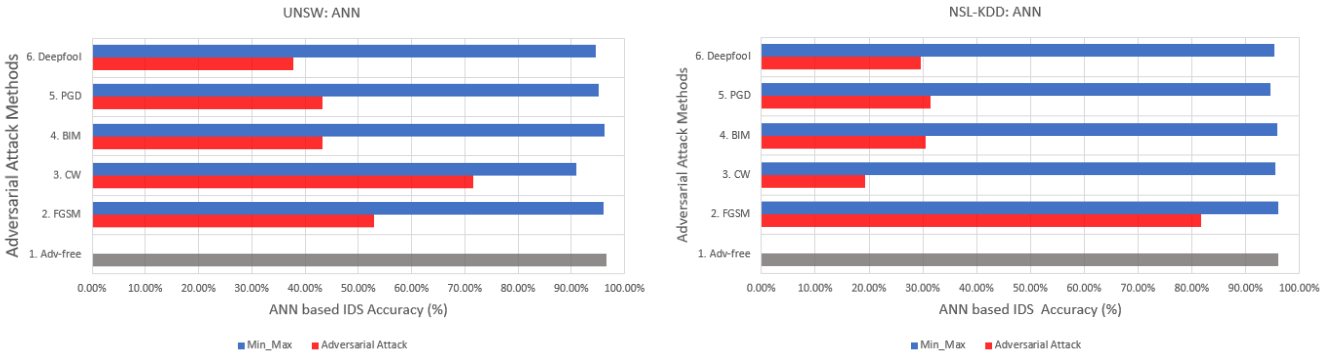
7.2 Phase II.2: Adversarial Training Experiments using min-max

In Phase II-2, we apply ***Algorithm I*** on ANN, CNN, and RNN to retrain the IDS models with the generated adversarial samples by inner-maximizer. We then evaluate the robustness of the defender adversarially trained: ANN, CNN, and RNN based IDSs on UNSW-NB15 and NSD-KDD with unseen adversarial samples. We also compare their performance to the baseline IDS models that we built in Phase I.

As a reminder, we have six baseline models built in Phase I. we refer to the three models trained on UNSW-NB15: ANN-UNSW IDS, CNN-UNSW IDS, RNN-UNSW IDS. We also refer to the three models trained on NSD-KDD: ANN-KDD IDS, CNN-KDD IDS, RNN-KDD IDS. In this Phase II-2, we built ten adversarially trained models for each ANN, CNN, and RNN algorithms: five adversarially trained models on UNSW-NB15 and five on NSD-KDD. In total, we have 30 adversarially trained models on top of the six baseline models. We refer to the trained models by the adversarial attack methods, as shown in Fig 6. For example, we call the IDS model trained by FGSM adversarial samples with the FGSM model.

Experimental results on UNSW-NB15 and NSD-KDD datasets show a significant improvement in the prediction accuracy of the IDS models among the ANN, CNN, and RNN. Figures 20, 21 and 22 compare the prediction accuracy on UNSW-NB15 and NSD-KDD for all deep learning-based IDSs after applying the min-max approach.

In general, when we retrain models using Algorithm I, we observe a steady increase in the accuracy of the models

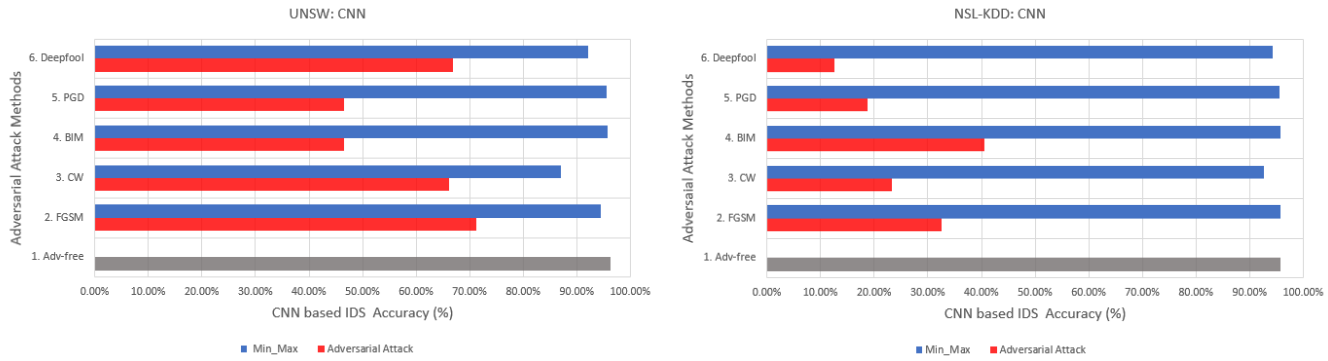


(a) UNSW-NB15 (b) NSD-KDD

Figure 20: Adversarial Attack vs Defense using min-max for ANN for NSD-KDD & UNSW-NB15

7.2.1 Performance of Adversarially Trained ANN IDS

As can be seen in Figure 20, the adversarially trained ANN on both UNSW-NB15 and NSD-KDD achieve high accuracy comparable to the baseline accuracy in Phase II-1. For example, ANN-FGSM models trained by FGSM adversarial samples from UNSW-NB15 and NSD-KDD are more robust to the generation of new FGSM adversarial samples. We notice in Phase II-1 a decrease of initial accuracy of the baseline ANN IDSs from 96.64% to 53% when the adversarial-free model attacked by FGSM samples, as shown in 20(a). After retraining the model using FGSM samples in the training phase using inner-maximization, the accuracy increased from 53% to 95.95%, which is almost the same as the adversarial-free model. Similarly, in the ANN-KDD model, all the adversarially trained models' accuracy increased, and the models correctly classified the adversarial samples. If we take the CW attack as an example in Figure 20(b), we can see how effective the CW samples against the adversarial-free model. CW has the most influence on accuracy. The accuracy decreased from 96.07% to 19.22% when CW attacks the ANN-KDD IDS model. When we retain the model with CW samples, the accuracy increased back to 95.99%.

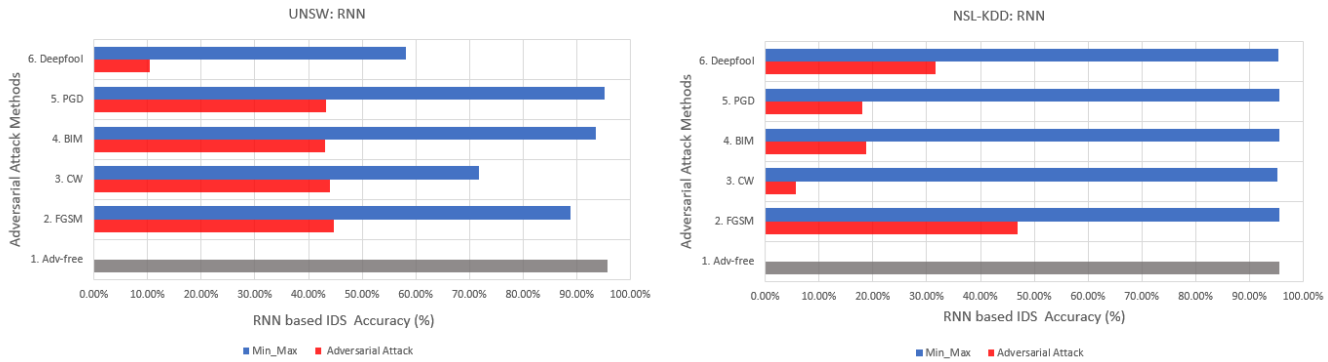


(a) UNSW-NB15 (b) NSD-KDD

Figure 21: Adversarial Attack vs Defense using min-max for CNN for NSD-KDD & UNSW-NB15

7.2.2 Performance of Adversarially Trained CNN IDS

As can be seen in Figure 21, the adversarially trained CNN on both UNSW-NB15 and NSD-KDD achieve high accuracy comparable to the baseline accuracy in Phase II-1. For example, CNN-FGSM models trained by FGSM adversarial samples from UNSW-NB15 and NSD-KDD are more robust to the generation of new FGSM adversarial samples. We notice in Phase II-1 a decrease of initial accuracy of the baseline CNN IDSs from 96.19% to 70.31% when the adversarial-free model attacked by FGSM samples, as shown in 21(a). After retraining the model using FGSM samples in the training phase using inner-maximization, the accuracy increased from 70.31% to 94.51%, almost the same as the adversarial-free model. Similarly, in the ANN-KDD model, all the adversarially trained models' accuracy increased, and the models correctly classified the adversarial samples. If we take a Deepfool attack as an example in Figure 21(b), we can see how effective the Deepfool samples against the adversarial-free model. Deepfool has the most influence on accuracy. The accuracy decreased from 95.69% to 12.61% when Deepfool attacks the CNN-KDD IDS model. When we retain the model with Deepfool samples, the accuracy increased back to 95.69%. Most of the adversarially trained models on both datasets are robust against all adversarial



(a) UNSW-NB15 (b) NSD-KDD

Figure 22: Adversarial Attack vs Defense using min-max for RNN for NSD-KDD & UNSW-NB15

attack samples. However, we can notice that CW samples' trained models were less robust to new CW samples, where accuracy achieved only 86.98% compared to other adversarial attacks.

7.2.3 Performance of Adversarially Trained RNN IDS

In this section, we evaluate the RNN IDS models after retraining them using Algorithm I. Figure 22 shows that the adversarially trained RNN IDSs on both UNSW-NB15 and NSD-KDD achieve high accuracy and outperform the baseline accuracy in Phase II-1. However, we can see that some of the adversarially trained RNN models, including Deepfool and CW models, seem not robust enough to new adversarial samples from the same attack methods. Both Deepfool and CW trained models to have a high miss classification rate where accuracy achieves only 58% against new Deepfool samples and 71% against new CW samples comparing to accuracy for FGSM, PGD and BIM trained models accuracy that achieves 88.83%, 95.23%, and 93.61% respectively. The nature of Deepfool attacks [12] makes it the most forceful attack methods.

After retraining the models in the training phase using inner-maximization, the accuracy for all adversarial models increased to almost 95% the same as the adversarial-free model accuracy, as shown in Figure 22.

Discussion

Relatively all adversarial trained models: ANN, CNN, and RNN trained using ***Algorithm I*** lead to high robustness against the five types of perturbations: FGSM, CW, BIM, PGD, and Deepfool generated by inner-maximizer.

- Superior results are seen when using ID framework based min-max when comparing our results in Phase II.1 The adversarially trained models are more robust to new adversarial samples and outperform the baseline IDSs.

- It is interesting to note that the relatively all of IDS models trained by CW samples are less robust to new CW samples comparing to other adversarial trained models. For example, in RNN-UNSW IDS, we observe that the CW model is less robust to new CW adversarial samples. ANN-UNSW IDS trained by CW accuracy is slightly less than other models. Similarly, for CNN-UNSW and CNN-KDD, the accuracy is less than other models. This result corresponds to the strong nature of the CW attack [55].

- Deepfool attack in RNN-UNSW has higher effectiveness compared to other attacks. We speculate that this might be due to the strong nature of Deepfool attack [12]

- However, even substantial results are achieved when using min-max. It must be pointed out that the accuracy of RNN-UNSW after applying adversarial training was the least robust comparing to other adversarial IDS models.

- FGSM, BIM, and PGD models were more robust to their adversarial samples.

Our experimental results show that min-max during the training with adversarial significantly improves the accuracy and the robustness of the adversarially trained models. With this behavior, we indicate adversarial training successfully solves the min-max optimization problem. Table 17 has all the experimental results of Phase II. In the next Chapter 8, we studied if the min-max optimization in deep learning-based IDS may consider a general defense against the maximum loss of multiple adversarial attacks.

Table 17: Accuracy Results (%) of Phase I and Phase II experiments on UNSW-NB15 and NSL-KDD datasets.

Dataset	Model	Adve-free	Attack Method	Adv. Attack Acc	min-max Acc
UNSW-NB15	ANN	96.64	FGSM	53	95.95
			CW	71.62	90.83
			BIM	43.27	96.21
			PGD	43.29	95.21
			DeepFool	37.87	94.57
	CNN	96.19	FGSM	71.31	94.51
			CW	66.20	86.98
			BIM	46.52	95.66
			PGD	46.45	95.47
			DeepFool	66.94	92.05
	RNN	95.71	FGSM	44.74	88.83
			CW	43.94	71.77
			BIM	43.15	93.61
			PGD	43.35	95.23
			DeepFool	10.36	58.18
NSL-KDD	ANN	96.07	FGSM	81.73	95.99
			CW	19.22	95.50
			BIM	30.49	95.87
			PGD	31.52	94.61
			DeepFool	29.60	95.32
	CNN	95.69	FGSM	32.48	95.69
			CW	23.37	92.67
			BIM	40.60	95.69
			PGD	18.71	95.52
			DeepFool	12.63	94.31
	RNN	95.47	FGSM	46.92	95.53
			CW	5.76	95.21
			BIM	18.71	95.54
			PGD	18.07	95.52
			DeepFool	31.59	95.32

Chapter 8

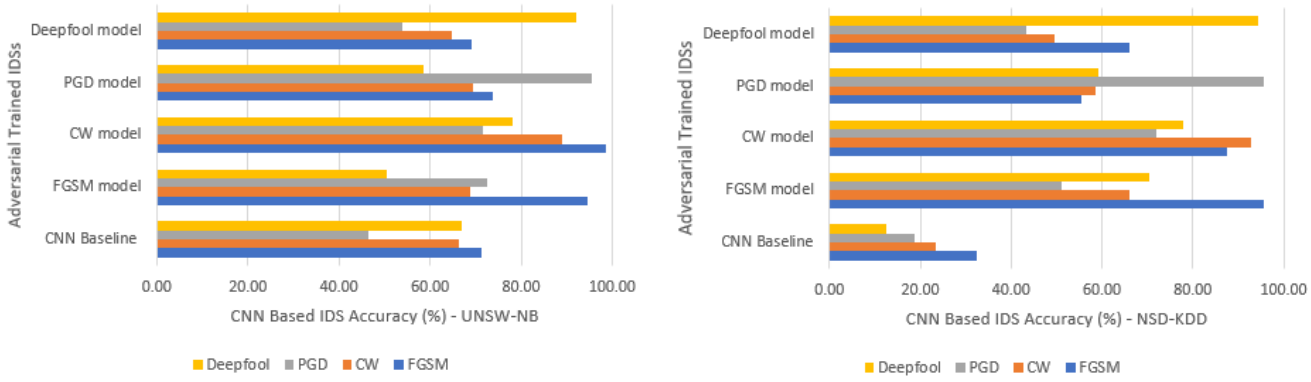
Phase III: Evaluation of Adversarial Trained IDSs against Multiple Adversarial Attacks

In this Chapter, we aim to study if the min-max optimization in deep learning-based IDS may consider a general defense against the maximum loss of multiple adversarial attacks. We investigate the effectiveness of multiple adversarial perturbation types during testing time against adversarial IDs trained via single perturbation type during training time.

We run the remaining experiments in Phase III using PHASE I experimental setup of CNN and RNN over UNSW-NB15 and NSD-KDD datasets in Chapter 5 and using four adversarial attacks methods. In this Chapter, we attack CNN and RNN models trained with a single type of attack during the training phase by four adversarial attacks to evaluate their robustness and compare the results with the four CNN and RNN baseline IDSs: CNN-UNSW, CNN-KDD, RNN-UNSW, and RNN-KDD. Selecting CNN and RNN architectures and not ANN, because, in our previous work in [1], we study the min-max on multiple adversarial attacks using ANN architecture. Also, in Chapter 6 and 7, we noticed that BIM and PGD attacks effectiveness in

UNSW-NB15 and NSD-KDD experiments in most of the models are nearly similar. Therefore, we decided to perform the remaining experiments in Phase III, using only four attacks, including FGSM, PGD, CW, and Deepfool.

In our analysis, we measure the accuracy before and after applying min-max formulation. We also want to compare the impact of adversarial samples generated by inner-maximization 2 on re-trained IDS and evaluate their robustness against multiple adversarial attacks during testing phase.



(a) CNN UNSW-NB15 (b) CNN NSD-KDD

Figure 23: Evaluation of multiple attacks against Adversarially Trained CNN models for NSD-KDD & UNSW-NB15

8.1 Performance of Adversarial Trained CNN IDSs

The performance of the two adversarial trained CNN based IDS 23(a) and 23(b) in adversarial environment is clear as shown in Figure 23. Evaluation of performance for the baseline IDS models in the adversarial environment is in Chapter 7. In this experiment, we attack the adversarially trained models that we trained in Phase II-2 with single type of perturbations by other adversarial attacks: FGSM, CW, PGD, and

Table 18: Prediction Accuracy Results for CNN-based IDS on UNSW-NB15. The model column indicates the trained models with a single type of adversarial via min-max. The adversarial attack methods row indicates the methods used to generate adversarial samples by the inner maximizer

-	Adversarial Attack Methods			
Model	FGSM	CW	PGD	Deepfool
CNN Baseline	**71.31**	**66.2**	**46.45**	**66.94**
FGSM model	**94.51**	68.82	72.76	50.42
CW model	98.69	**88.86**	71.6	78.05
PGD model	73.77	69.52	**95.47**	58.54
Deepfool model	69.07	64.66	53.9	**92.05**

Deepfool. Table 18 and 19 summarises the result of Phase III for the five adversarial trained CNN IDSs including the adversarial-free CNN IDS: CNN-UNSW and CNN-KDD. The result values of the two tables 18 and 19 are the prediction accuracy of five trained CNN IDSs.

Table 19: Prediction Accuracy Results for CNN-based IDS on NSD-KDD. The model column indicates the trained models with a single type of adversarial via min-max. The adversarial attack methods row indicates the methods used to generate adversarial samples by the inner maximizer

-	Adversarial Attack Methods			
Model	FGSM	CW	PGD	Deepfool
CNN Baseline	**32.48**	**23.37**	**18.71**	**12.63**
FGSM model	**95.69**	66.05	51.17	70.29
CW model	87.38	**92.67**	72.04	77.96
PGD model	55.36	58.43	**95.52**	59.13
Deepfool model	66.16	49.52	43.25	**94.31**

We observe, as we discussed in Chapter 7, the adversarial-free models (baselines) have low accuracy compared to the other adversarial trained CNN models. The

results are expected because we train the baseline models with adversarial-free and clean datasets, as seen in the bold values. We are also not surprised that all training methods are relatively robust to adversarial samples from the same attack methods, as seen in the bold and underlined values in Table 18 and Table 19.

As shown in Figure 23(a), the baseline accuracy of CNN-UNSW IDs decreased to 46% via PGD samples. By retraining the CNN-UNSW with PGD samples using min-max, the accuracy of the prediction increased to 95.47%. Similarly in Figure 23(a), the baseline accuracy of CNN-KDD IDs decreased to 18% via PGD samples. By retraining the CNN-KDD with PGD samples using min-max, the accuracy of the prediction increased to 95.52%. However, we noticed that some adversarial IDS models are more robust to other adversarial samples. For example, the CW model's accuracy is 98.69% against FGSM samples complying with CW samples accuracy that achieved only 88.86% against new CW samples, although the CW model was trained by CW samples not FGSM samples. We noticed that the FGSM model was resilient against CW attacks, and CW models were resilient to FGSM attacks. We speculate that this might be because the FGSM attack considers a fast attack not robust and generally has less effectiveness among all adversarial models.

CW model has the highest accuracy across all adversarial attack methods, where his overall accuracy average is almost 83%. FGSM model the second highest accuracy, where the average of all accuracy achieves almost 70%. PGD and Deepfool achieve 67% and 63% as an average accuracy.

If we take closer look to CNN-based IDS in both UNSW-NB15 and NSD-KDD, we find that PGD attack has the highest impact achieve 53.9% and 43.25 against Deepfool model in UNSW-NBB and NSD-KDD,respectively. While Deepfool attack accuracy is the lowest where it achieve to 50.42% against FGSM model. By comparing the results from Table 18 and 19 to determine which model more robust against

multiple adversarial attacks, we find CW models in both UNSW-NV15 and NSD-KDD accuracy achieved higher that 70% against all adversarial perturbations. We underlined CW model in both Tables 18 and 19.

In general, we noticed that model trained using min-max formulation is more robust to adversarial samples among all adversarial attack samples comparing to baseline models. When comparing our results to those of baseline Models in Figures 23(a) and 23(b), it must be pointed out that the min-max increases the robustness, especially in NSD-KDD.

A similar conclusion was reached in Chapter 7 about the datasets, We find that UNSW-NB models were more robust to adversarial environment comparing to NSD-KDD models in CNN-based IDS. It is difficult to explain such results within the context of evaluating different datasets, with different neural network architecture attacking with multiple adversarial attack methods.

8.2 Performance of Adversarial Trained RNN IDSs

In this experiment, we attack adversarial trained RNN IDSs with single adversarial attack type using Algorithm I by four adversarial attack methods: FGSM, CW, PGD, and Deepfool to evaluate the robustness of each adversarial model against multiple adversarial attacks.

Likewise adversarial trained CNN based IDSs, we noticed all models are more resilience to some adversarial samples than other adversarial samples.

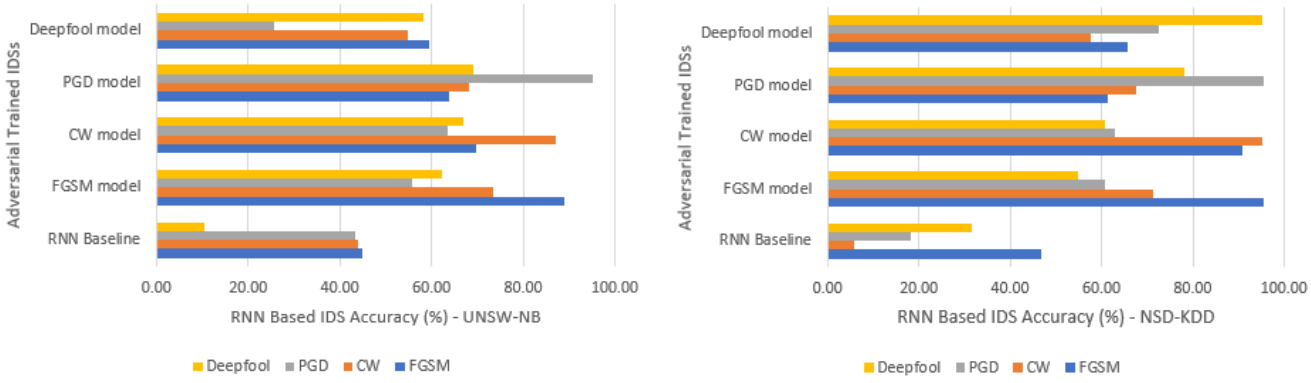
Figure 24 illustrates the prediction accuracy for the four adversarial trained models, including the RNN IDS baselines: RNN-UNSW and RNN-KDD. We can see in the figure that we have Figure 24(a) that represent all adversarial trained RNN IDSs

Table 20: Prediction Accuracy Results for RNN-based IDS on UNSW-NB15. The model column indicates the trained models with a single type of adversarial via min-max. The adversarial attack methods row indicates the methods used to generate adversarial samples by the inner maximizer

-	Adversarial Attack Methods			
Model	FGSM	CW	PGD	Deepfool
RNN Baseline	**44.74**	**43.94**	**43.35**	**10.36**
FGSM model	**88.83**	73.53	55.59	62.19
CW model	69.55	**87.09**	63.37	67.04
PGD model	63.67	68.22	**95.23**	69.13
Deepfool model	59.4	54.7	25.66	**94.75**

Table 21: Prediction Accuracy Results for RNN-based IDS on NSD-KDD. The model column indicates the trained models with a single type of adversarial via min-max. The adversarial attack methods row indicates the methods used to generate adversarial samples by the inner maximizer

-	Adversarial Attack Methods			
Model	FGSM	CW	PGD	Deepfool
RNN Baseline	**46.92**	**5.76**	**18.07**	**31.59**
FGSM model	**95.53**	71.30	60.85	54.93
CW model	90.95	**95.21**	62.99	60.78
PGD model	61.33	67.51	**95.52**	78.22
Deepfool model	65.58	57.55	72.66	**95.32**



(a) UNSW-NB15 (b) NSD-KDD

Figure 24: Prediction Accuracy of multiple attacks against Adversarially Trained RNN for NSD-KDD and UNSW-NB15

using the UNSW dataset, and Fig22(b) that represent all adversarial trained RNN IDSs trained by NSD-KDD dataset. We noticed that the adversarial-free models (baseline) have low accuracy compared to the other adversarial trained RNN models. The results are expected because the baseline models are trained by an adversarial-free and clean dataset, as seen in the bold values. We are also not surprised that all training methods are relatively robust to adversarial samples from the same adversarial attack methods, as shown in the bold and underlined values in Table 21 and 20.

For example, as shown in Figure 22(a), the baseline accuracy of RNN-UNSW IDs decreased to 44% via FGSM samples. By retraining the RNN-UNSW with FGSM samples using min-max, the accuracy of the prediction increased to 88.83%.

If we take closer look at both Tables 20 and 21, we find that Deepfool attack is robust where accuracy achieves to 54.93% against FGSM model. This accuracy is consistent with what has been found in CNN-UNSW, where accuracy in the FGSM model achieves 50.42%

In general, we noticed that model trained using min-max formulation is more robust to adversarial samples among all adversarial attack samples comparing to

baseline models. When comparing our results to those of baseline Models in Figures 24(a) and 24(b), it must be pointed out that the min-max increases the robustness, especially in NSD-KDD. Approximately, accuracy in all models in Figure 24(b) and 24(a) increased more than 60%.

A similar conclusion was reached in Chapter 7 about the datasets, We find that UNSW-NB models were roughly more robust to adversarial environment comparing to NSD-KDD models in RNN-based IDS. It is difficult to explain such results within the context of evaluating different datasets, with different neural network architecture attacking multiple adversarial attack methods.

Discussion

Through the comprehensive comparison and analysis of the Phase III experimental results, we can draw the following conclusions about the robustness of the adversarial trained IDSs:

- Deepfool model's overall architectures and datasets have the least robustness against all four adversarial attack methods. More specifically, the Deepfool model robustness in RNN-UNSW was the least compared to other models.

- CW models overall architectures and datasets have the best robustness against all four adversarial attack methods. More specifically, the accuracy of CNN-UNSW and CNN-KDD, when trained by CW samples, outperform other adversarial trained models. We noticed that some adversarial IDS models are more robust to other adversarial samples. Like the CW model's accuracy is 98.69% against FGSM samples complying with CW samples accuracy that achieved only 88.86% against new CW samples, although the CW model was trained by CW samples not FGSM samples. We speculate that this might be because the

FGSM attack considers a fast attack not robust and generally has less effectiveness among all adversarial models.

- FGSM models have the same robustness against all adversarial attack methods in both datasets in CNN and RNN.

- PGD models robustness has the same robustness as FGSM models against all adversarial attacks overall architectures and datasets.

- The thorough experiments show that Algorithm 4.2 based min-max optimization leads to substantially improvement among attacking using multiple state-of-the-art adversarial attacks compared with the baseline IDS models accuracy results in Phase II.1 in Chapter 7. The overall adversarially IDS models are robust to all adversarial attacks. However, the results demonstrated in this Chapter did not match the result in Phase II.2 in Chapter 7 where adversarial attacks was using single type of attack in training and testing time.

  The min-max approach still does not represent a universal cure for different adversarial attacks attacking adversarially trained model with one type of adversarial samples.

Chapter 9

Conclusions

9.1 Summary

In this research, we attempt to study adversarial training, which models the problem based on min-max (or saddle-point) optimization as an attack and defense against adversarial perturbations techniques in an adversarial environment to increase robustness. The Novelty of this research derives from the fact that it is the first experiment that implements and compares different types of neural networks architectures, e.g., ANN, CNN, and RNN, in deep learning-based IDS in an adversarial environment, and evaluated on two IDS benchmark datasets: UNSW-NB15 and NSD-KDD using various adversarial attack methods. Most of the literature in the adversarial domain demonstrated the concept of using one deep learning architecture [1] [90] with different adversarial attacks [52] [23]. This research took the path further in investigating and evaluating the robustness of different types of DNN based IDS against five state-of-the-art adversarial attack methods: FGSM, CW, BIM, PGD, and Deepfool using the min-max formulation. We spent significant effort in building 36 Deep learning-based IDS and conduct comprehensive verification to prove that the adversarial training based min-max formulation considers a strong defense technique against different adversarial samples.

To do that, we utilize five well-known adversarial attack methods to generate persuasive adversarial samples that maximize the loss using the inner maximization and inject them during the training time to minimize the loss. We divide our experiments into three phases, and we utilize three different deep learning algorithms: ANN, CNN, RNN, to train six baseline models in Phase I and Phase II on two benchmark datasets: UNSW-NB15 and NSD-KDD. We split the two datasets into training, testing, and validation sets and then preprocessing them. Part of the experiments that we conduct is to study and analyze the effect of feature selection on two IDS benchmark datasets: UNSW-NB15 and NSD-KDD. We apply feature selection using two techniques: PCA and Recursive Feature Elimination (RFE), and RFE results are higher comparing to PCA. We selected the RFE feature selection to move forward with the entire experiments in phase II and phase III. At the end of Phase I, we build highly accurate deep learning models on UNSW-NB15 and NSD-KDD for each architecture: ANN, CNN, and RNN as baseline models (benchmarks) to comprehensively compare the performance of Deep learning-based IDSs in the adversarial environment in Phases II. In Phase II, we then utilize well-know adversarial attack methods to generate adversarial samples: FGSM, BIm, PGD, CW, and Deepfool using the inner-maximization problem. We then used outer-minimization to perform the adversarial training process based on the min-max approach. In Phase III, we evaluate min-max formulation among the robustness of adversarial models retrained with a single adversarial perturbation method against multiple adversarial perturbation methods.

To summarize this work, we will answer various questions, including the contributions, results, and findings, and what aspect is still unanswered.

1. The results confirm that RFE is a good choice for feature selection for both UNSW-NB15 and NSD-KDD. Deep learning-based IDS models achieve high accuracy in the three architectures. ANN-based IDS accuracy achieves 95.71% and 95.47% for UNSW-NB15 and NSD-KDD, respectively. CNN based IDS

accuracy achieves 96.19% and 95.69% for UNSW-NB15 and NSD-KDD, respectively. RNN based IDS accuracy achieves 96.64% and 96.06% for UNSW-NB15 and NSD-KDD, respectively. The Confusion matrix and ROC curve show that various matrices, including precision, recall, and F-score, show how accurately the IDSs can predict. All three types of deep learning consider best to classify UNSW-NB15, and NSD-KDD traffic flow as all of them have almost similar accuracy.

2. We generated persuasive adversarial samples from the five well-known adversarial attack methods with high perturbation using inner-maximizer. We then evaluate the quality of the adversarial samples found on the UNSW-NB15 and NSK-KDD datasets against the six baseline IDS models. Our results demonstrated that:

   FGSM attack among all deep learning-based IDS has less impact on almost all baseline models because of the nature of the FGSM attack.

   BIM and PGD attacks effectiveness in UNSW-NB15 and NSD-KDD experiments in most of the models are nearly similar, and the difference minimal and difficult to observe.

   CW and Deepfool attacks results of the experiments found clear support for the previous finding in other research that both attacks consider potent attacks and can generate small perturbation to evade deep learning models.

3. We retrain the IDS models using min-max and compare their performance to the baseline IDS models. This yields increasingly superior results on IDS performance in an adversarial environment. The results are equal to or better than a result obtained in the initial training using an adversarial-free dataset. The results of attacking the models that are adversarially trained using min-max are substantially better than those of baseline IDS models trained with

adversarial-free datasets.

4. At last, we investigated the vulnerability of adversarially trained deep learning-based IDS with a single type of perturbation to adversarial perturbations from multiple attack methods. We aimed to investigate if the min-max optimization considers a general defense approach. We attacked CNN and RNN adversarially trained model by four adversarial attacks: FGSM, CW, PGD, and Deepfool to evaluate their robustness and compare the results. The results are substantially better than the result of attacking the baseline IDS models in Phase II.1 in Chapter 7, where the overall adversarially IDS models are robust to all adversarial attacks. The overall accuracy did not decrease more than 50% of the initial accuracy. However, following the results of this phase, we can conclude that the min-max approach still does not represent a universal cure for different adversarial attacks attacking an adversarially trained model with one single type adversarial samples.

9.2 Future Work

Although adversarial training-based min-max approach considers a countermeasure and a robust design against state-of-art adversarial attacks, it still does not represent a universal cure for different adversarial attacks attacking adversarially trained model with one type of adversarial samples. Therefore, a universal defense and solution remain an open problem in deep learning-based IDS. We believe exploration in the Transferable Adversarial Training approach [87] by generating transferable adversarial samples and augmenting them during training time in other models in the intrusion detection domain might lead to optimization solutions in an adversarial environment.

List of References

[1] R. Abou Khamis, O. Shafiq, and A. Matrawy, "Investigating resistance of deep learning-based ids against adversaries using min-max optimization," *IEEE ICC 20*, 2020.

[2] R. Abou Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss," *arXiv preprint arXiv:2007.04472*, 2020.

[3] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[4] Neptune.ai, "F1 score vs roc auc vs accuracy vs pr auc: Which evaluation metric should you choose?."

[5] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[7] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach," *ICST Trans. Security Safety*, 2018.

[8] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *ICLR (Workshop)*, 2014.

[9] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples (2014)," *arXiv preprint arXiv:1412.6572*.

[11] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *arXiv preprint arXiv:1712.07107*, 2017.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[13] U. Jang, X. Wu, and S. Jha, "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 262–277, 2017.

[14] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 39–49, ACM, 2017.

[15] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25, ACM, 2006.

[16] A. Nayebi and S. Ganguli, "Biologically inspired protection of deep networks from adversarial attacks," *arXiv preprint arXiv:1703.09202*, 2017.

[17] E. Viegas, A. Santin, N. Neves, A. Bessani, and V. Abreu, "A resilient stream learning intrusion detection mechanism for real-time analysis of network traffic," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35, IEEE, 2018.

[19] L. Chen, Y. Ye, and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *Intelligence and Security Informatics Conference (EISIC), 2017 European*, pp. 99–106, IEEE, 2017.

[20] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," *CoRR*, 2018.

[21] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, 2017.

[22] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, pp. 5286–5295, 2018.

[23] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 76–82, IEEE, 2018.

[24] O. Ibitoye, R. Abou-Khamis, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security–a survey," *arXiv preprint arXiv:1911.02621*, 2019.

[25] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[26] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011.

[27] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[28] F. Kuang, S. Zhang, Z. Jin, and W. Xu, "A novel svm by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection," *Soft Computing*, vol. 19, no. 5, pp. 1187–1199, 2015.

[29] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm," in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, pp. 181–186, IEEE, 2017.

[30] H. A. Mahmood, "Network intrusion detection system (nids) in cloud environment based on hidden naïve bayes multiclass classifier," *Al-Mustansiriyah Journal of Science*, vol. 28, no. 2, pp. 134–142, 2018.

[31] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, IEEE, 2016.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[33] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2016.

[34] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 258–263, IEEE, 2016.

[35] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21954–21961, 2017.

[36] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[37] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[38] L. Mohammadpour, T. C. Ling, C. S. Liew, and C. Y. Chong, "A convolutional neural network for network intrusion detection system," *Proceedings of the Asia-Pacific Advanced Network*, vol. 46, pp. 50–55, 2018.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South African Computer Journal*, vol. 56, no. 1, pp. 136–154, 2015.

[41] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2157–2169, 2017.

[42] P. Laskov and R. Lippmann, "Machine learning in adversarial environments," 2010.

[43] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli, "Secure kernel machines against evasion attacks," in *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pp. 59–69, ACM, 2016.

[44] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *CoRR*, 2018.

[45] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*, pp. 137–149, Springer, 2016.

[46] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 533–537, IEEE, 2018.

[47] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, 2016.

[48] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2014.

[49] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[50] C. Burkard and B. Lagesse, "Analysis of causative attacks against svms learning from data streams," in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, pp. 31–36, ACM, 2017.

[51] F. Zhang, P. P. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE transactions on cybernetics*, vol. 46, no. 3, pp. 766–777, 2016.

[52] Y. Peng, J. Su, X. Shi, and B. Zhao, "Evaluating deep learning based network intrusion detection system in adversarial environment," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 61–66, IEEE, 2019.

[53] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *computers & security*, vol. 73, pp. 326–344, 2018.

[54] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[55] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, 2017.

[56] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[57] Z. Yan, Y. Guo, and C. Zhang, "Deep defense: Training dnns with improved adversarial robustness," in *Advances in Neural Information Processing Systems*, pp. 419–428, 2018.

[58] G. K. Santhanam and P. Grnarova, "Defending against adversarial attacks by leveraging an entire gan," *arXiv preprint arXiv:1805.10652*, 2018.

[59] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[60] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[61] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *ICLR*, 2018.

[62] O. Ibitoye *et al.*, "Analyzing adversarial attacks against deep learning for intrusion detection in iot networks," *IEEE GlobalCom*, 2019.

[63] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 405–412, 2010.

[64] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2013.

[65] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," *arXiv preprint arXiv:1904.02144*, vol. 3, 2019.

[66] K. Grosse, D. Pfaff, M. T. Smith, and M. Backes, "The limitations of model uncertainty in adversarial settings," *arXiv preprint arXiv:1812.02606*, 2018.

[67] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

[68] F. Khalid, M. A. Hanif, S. Rehman, J. Qadir, and M. Shafique, "Fademl: understanding the impact of pre-processing noise filtering on adversarial machine learning," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 902–907, IEEE, 2019.

[69] M. Usama, M. Asim, S. Latif, J. Qadir, *et al.*, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 78–83, IEEE, 2019.

[70] M. Cheng, Q. Lei, P.-Y. Chen, I. Dhillon, and C.-J. Hsieh, "Cat: Customized adversarial training for improved robustness," *arXiv preprint arXiv:2002.06789*, 2020.

[71] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?," in *International Conference on Machine Learning*, pp. 1689–1698, 2015.

[72] Y. Hou, Z. Chen, M. Wu, C.-S. Foo, X. Li, and R. M. Shubair, "Mahalanobis distance based adversarial network for anomaly detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3192–3196, IEEE, 2020.

[73] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, 2020.

[74] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[75] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[76] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Computers & Security*, vol. 77, pp. 304–314, 2018.

[77] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2017.

[78] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 195–200, IEEE, 2016.

[79] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, vol. 239, pp. 201–225, 2013.

[80] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj, "Model evasion attack on intrusion detection systems using adversarial machine learning," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2020.

[81] A. Shafahi, M. Najibi, Z. Xu, J. P. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training.," in *AAAI*, pp. 5636–5643, 2020.

[82] Y. Balaji, T. Goldstein, and J. Hoffman, "Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets," *arXiv preprint arXiv:1910.08051*, 2019.

[83] J. Wang, T. Zhang, S. Liu, P.-Y. Chen, J. Xu, M. Fardad, and B. Li, "Towards a unified min-max framework for adversarial exploration and robustness," *arXiv preprint arXiv:1906.03563*, 2019.

[84] H. Zhang, H. Chen, Z. Song, D. Boning, I. S. Dhillon, and C.-J. Hsieh, "The limitations of adversarial training and the blind-spot attack," *arXiv preprint arXiv:1901.04684*, 2019.

[85] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195–204, 2018.

[86] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[87] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[88] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang, "Max-margin adversarial (mma) training: Direct input space margin maximization through adversarial training," *arXiv preprint arXiv:1812.02637*, 2018.

[89] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, pp. 1885–1893, 2016.

[90] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *International workshop on multiple classifier systems*, pp. 350–359, Springer, 2011.

[91] M.-I. Nicolae *et al.*, "Adversarial robustness toolbox v0. 2.2," *arXiv preprint arXiv:1807.01069*, 2018.

[92] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[93] S. Revathi and A. Malathi, "A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 1848–1853, 2013.

[94] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *2015 international conference on signal processing and communication engineering systems*, pp. 92–96, IEEE, 2015.

[95] H.-s. Chae, B.-o. Jo, S.-H. Choi, and T.-k. Park, "Feature selection for intrusion detection using nsl-kdd," *Recent advances in computer science*, pp. 184–187, 2013.

[96] S. Narkhede, "Understanding confusion matrix."

[97] J. W. Ulvila and J. E. Gaffney Jr, "Evaluation of intrusion detection systems," *Journal of Research of the National Institute of Standards and Technology*, vol. 108, no. 6, p. 453, 2003.

[98] J. N. Mandrekar, "Receiver operating characteristic curve in diagnostic test assessment," *Journal of Thoracic Oncology*, vol. 5, no. 9, pp. 1315–1316, 2010.

[99] Y. Meng, "Measuring intelligent false alarm reduction using an roc curve-based approach in network intrusion detection," in *2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings*, pp. 108–113, IEEE, 2012.

[100] X. Liu, D. Klabjan, and P. NBless, "Data extraction from charts via single deep neural network," *arXiv preprint arXiv:1906.11906*, 2019.

[101] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

Appendix A

Table 22: UNSW-NB1515 15 dataset Features based on the description in [5]

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO,, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes

10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the row packet size transmitted by the src
24	dmeansz	integer	Mean of the row packet size transmitted by the dst
25	trans$_d$*epth*	integer	Represents the pipelined depth into the connection of http request/response transaction

26	res$_b dy_l en$	integer	Actual uncompressed content size of the data transferred from the server's http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYNACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYNACK and the ACK packets.
36	is$_s m_i ps_p orts$	Binary	If source (1) and destination (3) IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct$_s tate_t tl$	Integer	No. for each state
38	ct$_f lw_h ttp_m thd$	Integer	No. of flows that has methods such as Get and Post in http service.

39	is$_{f}tp_{l}ogin$	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct$_{f}tp_{c}md$	integer	No of flows that has a command in ftp session.
41	ct$_{s}rv_{s}rc$	integer	No. of connections that contain the same service
42	ct$_{s}rv_{d}st$	integer	No. of connections that contain the same service
43	ct$_{d}st_{l}tm$	integer	No. of connections of the same destination address
44	ct$_{s}rc_{l}tm$	integer	No. of connections of the same source address
45	ct$_{s}rc_{d}port_{l}tm$	integer	No of connections of the same source address
46	ct$_{d}st_{s}port_{l}tm$	integer	No of connections of the same destination address
47	ct$_{d}st_{s}rc_{l}tm$	integer	No of connections of the same source .
48	attack$_{c}at$	nominal	The name of each attack category.
49	Label	binary	0 for normal and 1 for attack records

Table 23: NSL-KDD Attack Type

Major Attacks	Type
DOS	Back
	Land,Neptune
	Pod, Smurf
	Teardrop,Mailbomb
	Processtable,Udpstor
	Apache2, Worm
R2L	Guesspassword
	Ftpwrite
	Imap,Phf
	Multihop,Warezmaster
	Xlock,Xsnoop
	Snmpguess,Snmpgetattack
	Httptunnel,Sendmail
	Named
U2L	Bufferoverflow
	Loadmodule
	Rootkit
	Perl
	Sqlattack
	Xterm,Ps
Probe	Satan
	IPsweep, Nmap
	Portsweep,Mscan
	Sa,int

Appendix B

B.1 Results

Table 24: Accuracy Rate Results of adversarial attacks in ANN. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	ANN	96.64	53	71.62	43.27	43.27	37.87
NSLKDD	ANN	96.07	81.73	19.22	30.49	31.52	29.60

Table 25: Accuracy Rate Results of Adversarial Training for ANN Experiments using min-max. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	ANN	96.64	95.95	90.83	96.21	96.21	94.57
NSLKDD	ANN	96.07	95.99	95.59	95.87	94.61	95.32

Table 26: Accuracy Rate Results of adversarial attacks in CNN. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	CNN	96.19	71.31	66.20	46.52	46.52	66.94
NSLKDD	CNN	95.69	32.48	23.37	49.60	18.71	12.63

Table 27: Accuracy Rate Results of Adversarial Training for CNN Experiments using min-max. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	CNN	96.19	94.51	86.98	95.66	95.47	92.05
NSLKDD	CNN	95.69	95.69	92.67	95.69	95.52	94.31

Table 28: Accuracy Rate Results of adversarial attacks in RNN. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	RNN	95.47	46.92	5.76	18.71	18.71	31.59
NSLKDD	RNN	95.69	44.74	43.94	43.15	43.15	10.36

Table 29: Accuracy Rate Results of Adversarial Training for RNN Experiments using min-max. The adversarial attack methods row indicates the methods used that generate adversarial samples by the min-max method

Dataset	Model	Natural	FGSM	CW	BIM	PGD	Deepfool
UNSWNB	RNN	96.19	94.51	86.98	95.66	95.47	92.05
NSLKDD	RNN	95.69	95.69	92.67	95.69	95.52	94.31