NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TOWARDS ROBUST LEARNING USING DIAMETRICAL RISK MINIMIZATION FOR NETWORK INTRUSION DETECTION

by

Kelson J. McCollum

June 2023

Thesis Advisor: Johannes O. Royset
Second Reader: Nathaniel Bastian,
Army Cyber Institute

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE TOWARDS ROBUST LEARNING USING DIAMETRICAL RISK MINIMIZATION FOR NETWORK INTRUSION DETECTION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Kelson J. McCollum	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Currently, deep neural networks (DNNs) show great promise in the detection of malicious network traffic at machine speed. However, these networks are typically trained using Empirical Risk Minimization (ERM), which is not robust to misclassified or altered training data. We propose applying Diametrical Risk Minimization (DRM), which is shown to lead to more robust optimization solutions, to train DNNs to classify malicious network traffic. Using two different network traffic datasets, we find that when state-of-the-art DNNs are trained on partially mislabeled data, utilizing DRM results in higher accuracy compared to equivalent models trained with ERM in 13 of 20 cases examined, with ERM being more accurate in only 5 of the 20 cases. More importantly, when models are tested against previously unseen cyber-attack types, models trained with DRM correctly identify the previously unseen cyber-attacks more often. Of the 46 cases we examine, models trained with DRM show better performance compared to models trained with ERM in 25 cases and equal performance in an additional 10 cases. We show that these DNNs are computationally tractable to deploy in real-time on edge computing systems utilizing commercial-off-the-shelf hardware.

14. SUBJECT TERMS network intrusion detection, machine learning, robust learning, diametrical risk minimization	**15. NUMBER OF PAGES** 61
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

TOWARDS ROBUST LEARNING USING DIAMETRICAL RISK MINIMIZATION FOR NETWORK INTRUSION DETECTION

Kelson J. McCollum
Captain, United States Marine Corps
BS, United States Naval Academy, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2023

Approved by: Johannes O. Royset
Advisor

Nathaniel Bastian
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Currently, deep neural networks (DNNs) show great promise in the detection of malicious network traffic at machine speed. However, these networks are typically trained using Empirical Risk Minimization (ERM), which is not robust to misclassified or altered training data. We propose applying Diametrical Risk Minimization (DRM), which is shown to lead to more robust optimization solutions, to train DNNs to classify malicious network traffic. Using two different network traffic datasets, we find that when state-of-the-art DNNs are trained on partially mislabeled data, utilizing DRM results in higher accuracy compared to equivalent models trained with ERM in 13 of 20 cases examined, with ERM being more accurate in only 5 of the 20 cases. More importantly, when models are tested against previously unseen cyber-attack types, models trained with DRM correctly identify the previously unseen cyber-attacks more often. Of the 46 cases we examine, models trained with DRM show better performance compared to models trained with ERM in 25 cases and equal performance in an additional 10 cases. We show that these DNNs are computationally tractable to deploy in real-time on edge computing systems utilizing commercial-off-the-shelf hardware.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

2 Background and Methodology **3**
2.1 Literature Review 3
2.2 Methodology 7

3 Computational Experimentation **17**
3.1 The γ Neighborhood 17
3.2 Initial Performance Comparisons 19

4 Model Performance and Operational Considerations **23**
4.1 Deep Neural Network Results 23
4.2 Is Diametrical Risk Minimization Generalizing Better? 33
4.3 Operational Considerations 34

5 Conclusion **35**
5.1 Future Work 36

List of References **37**

Initial Distribution List **41**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 2.1 Sharp vs. Flat Minimums 4

Figure 2.2 ResNet Solutions 5

Figure 2.3 Network Packet 9

Figure 2.4 Fully-Connected Neural Network Architecture 10

Figure 2.5 One-Dimensional Convolutional Neural Network Architecture . . 11

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 2.1 CICIDS2017 Tuning Data Subset 12

Table 2.2 UNSW-NB15 Tuning Data Subset 13

Table 2.3 CICIDS2017 Full-Scale Data Subset 14

Table 2.4 UNSW-NB15 Full-Scale Data Subset 15

Table 3.1 Preliminary Clean Data Results 19

Table 3.2 Preliminary Mislabeled Data Results 20

Table 3.3 Preliminary Withheld Attack Results 21

Table 4.1 Mislabeled Data Results 23

Table 4.2 CICIDS2017 Withheld Attack Results 26

Table 4.3 UNSW-NB15 Withheld Attack Results 30

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

1dCNN One-Dimensional Convolutional Neural Network

AMP Adversarial Model Perturbation

AUC Area Under the Curve

COTS Commercial-Off-the-Shelf

DDoS Distributed Denial of Service

DNN Deep Neural Network

DOD Department of Defense

DoS Denial of Service

DRM Diametrical Risk Minimization

ERM Empirical Risk Minimization

FcNN Fully Connected Neural Network

FPR False Positive Rate

FTP File Transfer Protocol

IP Internet Protocol

ML Machine Learning

NIDS Network Intrusion Detection System

NN Neural Network

PCAP Packet Capture

PPV Positive Predictive Value

ReLU Rectified Linear Unit

SAM Sharpness-Aware Minimization

SGD Stochastic Gradient Descent

SQL Structured Query Language

SSH Secure Shell

TNR True Negative Rate

TPR True Positive Rate

Executive Summary

The Department of Defense (DOD), through priorities such as the U.S. Marine Corps' Force Design 2030 and the U.S. Navy's Force Design 2045, has indicated that networked unmanned combat systems are an integral component of both modern-day and future conflict. Significant work is being conducted to design and develop these systems to help the U.S. military achieve overmatch on the battlefield. However, these unmanned combat systems are vulnerable to increasingly sophisticated cyber-attacks being developed by our adversaries every day. Operating an unmanned system in an operational environment characterized by jamming, network congestion, and outages poses a unique risk in that a human operator may not always be in the loop to ensure nominal system behavior and performance.

Significant work has been conducted to apply machine learning (ML) via deep neural networks (DNNs) in an effort to detect and respond to cyber-attacks at machine speeds. This thesis extends that work by applying recently developed robust learning techniques to train state-of-the-art DNNs that are computationally tractable in real time on edge computing systems utilizing commercial-off-the-shelf (COTS) hardware. Our goal is to develop Network Intrusion Detection System (NIDS) models that are accurate, robust to mislabeled cyber-attack training data; generalize well to previously unseen cyber attacks, or zero-day exploits; and can be deployed on unmanned combat systems at the tactical edge of the battlespace without a human operator in the loop.

In this work, we utilize two state-of-the-art DNN architectures: a fully connected neural network (FcNN) and a one-dimensional convolutional neural network (1dCNN). These architectures operate by examining only the raw packet data that each edge computing system is receiving; they require no human in the loop and no knowledge of overall network flows. We train these networks using two datasets commonly cited in literature that contain both benign and malicious network traffic. We train utilizing two different optimization approaches: Empirical Risk Minimization (ERM), which is currently used in these types of models but is generally not robust to mislabeled or altered training data, and Diametrical Risk Minimization (DRM), which is shown to lead to more robust optimization solutions.

We find that across dataset and model architecture, models trained with the recently de-

veloped DRM are more accurate than models trained with ERM when trained on partially mislabeled data (the kind of data we might expect to see in real-world implementation). DRM is more accurate than ERM in 13 of the 20 cases we examine, while ERM is shows greater accuracy than DRM in 5 cases. With realistic amounts of mislabeling present, these models show test set accuracy rates greater than 98%.

Further, we find that when tested against previously unseen cyber-attacks, models trained with DRM are consistently better at catching the novel attack type. Of the 46 cases we examine, models trained with DRM show better performance compared to models trained with ERM in 25 cases and equal performance in an additional 10 cases.

The better test performance of DRM comes at the cost of additional computational complexity during model training, with DRM trained models taking an average of 1.7x longer to train than their ERM counterparts. However, choice of DNN architecture is actually much more impactful than the choice to use DRM. The more complex 1dCNN architecture takes on average 4.8x longer to train compared to an equivalent FcNN.

Importantly, when it comes to deployment there is no speed difference between classifying network traffic using models trained with ERM or DRM. We show that utilizing COTS hardware to run these DNNs is not only computationally tractable, but it can be done at relevant speeds. On an Apple M1 chip we can process and then classify incoming packets at a rate of 25.4 MBps utilizing a FcNN and 9.4 MBps with a 1dCNN.

We conclude by recommending that if the additional compute resources are available, DNNs be trained utilizing DRM due to the increased robustness to mislabeled training data and previously unseen attack types that DRM trained models exhibit. This thesis demonstrates a viable solution to help cyber harden current and future unmanned combat systems across the DOD. Importantly, our solution is accurate regarding classification of benign and malicious network traffic, robust to the amounts of mislabeling we might expect in real-world data, and exhibits improved performance against novel cyber-attack types. Further, it is computational tractable when deployed to examine every incoming packet in real time on an edge computing system utilizing COTS hardware, without requiring a human operator in the loop.

Acknowledgments

This thesis would not have been possible without the help and support of so many people around me. First and foremost my wife, Brooke. Thank you for all your understanding and support throughout this process, from the late nights to me checking my laptop in the middle of dinner (you know those models aren't going to babysit themselves...). To my parents, thank you for your interest and encouragement during not just this process, but throughout my entire education.

To my cohort, who I have spent countless hours with over the course of this program, thank you. From our productive study sessions to the occasional afternoon of relaxation on the golf course, the support you have shown has been extraordinary. We are all better and smarter for it.

Finally to my advisor team, Dr. Johannes Royset and LTC Nathaniel Bastian, thank you for sharing your knowledge, expertise, and time with me over the past year. From the sheets of handwritten algorithms to complex code walkthroughs, I am truly grateful for not just your assistance, but also your mentorship throughout this process. This thesis truly would not have been possible without the two of you.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

From 2015 to 2021, the U.S. Department of Defense (DOD) experienced over 12 thousand cyber attacks (GAO 2022). The true number is unknown, because many attacks go undetected. For example, in May of 2023 the U.S. discovered a cyber-attack targeting critical DOD infrastructure in in the Pacific; that attack had been ongoing since mid-2021 (Microsoft Threat Intelligence 2023).

Clearly, the prevalence of cyber-attacks combined with the increased networking and connectivity of DOD and commercial systems poses an acute security risk both before and during conflict. A solution is needed to reliably identify malicious network traffic as it comes in to, but before it is processed by networked hardware, without a human operator in the loop. Any proposed Network Intrusion Detection System (NIDS) must be accurate regarding classification of malicious and benign traffic, robust to mislabeling of training data, and able to recognize modifications or extensions of known cyber-attacks.

Especially pertinent to a military use case is the cybersecurity of unmanned systems. Increasingly, unmanned systems are being employed at the tactical edge of the battlespace. These systems may not always have a human operator monitoring their performance, especially in the presence of communications and datalink jamming likely to be ubiquitous in any future near-peer conflict. These systems must be cyber hardened, and come with the additional complication that computing power is likely limited at the edge. Therefore, any proposed NIDS must be computationally tractable in real time on edge computing systems.

Currently, machine learning (ML) shows great promise in automating the detection of cyber-attacks with a high degree of accuracy utilizing deep neural networks (DNNs) (De Lucia et al. 2021; Bierbrauer et al. 2023). However, modern DNN architectures tend to overfit to the training data (Zhang et al. 2021) and therefore do not generalize well in the presence of mislabeled training data. Diametrical Risk Minimization (DRM), also known as Sharpness-Aware Minimization (SAM), or Adversarial Model Perturbation (AMP), is a technique recently developed (Norton and Royset 2021; Foret et al. 2020; Zheng et al. 2021) that seeks to improve the generalization performance of DNNs. In real-world applications,

cyber-attack training data for ML algorithms will likely not be perfectly clean. Additionally, new exploits will be discovered on a regular basis, leading to novel types of cyber-attacks. We hypothesize training DNNs using DRM for network intrusion detection will result in improved robustness.

In this thesis, we evaluate the impact of training two DNN architectures using DRM as opposed to the currently used Empirical Risk Minimization (ERM) to determine whether DRM can improve on current state-of-the-art performance. We also examine whether training a DNN using DRM has the potential to provide a ML-based NIDS that exhibits greater robustness to mislabeled training data and novel cyber-attack types.

The remainder of this thesis is organized as follows. Chapter 2 explores the background of DRM and applications of ML to network intrusion detection, then describes the methodology for combining these two areas of research. Chapter 3 presents our computational experimentation for DNN model training using DRM. Chapter 4 continues with performance comparisons of ERM and DRM and a discussion of computational tractability. Finally, Chapter 5 provides a summary of findings and recommendations for future work.

CHAPTER 2:
Background and Methodology

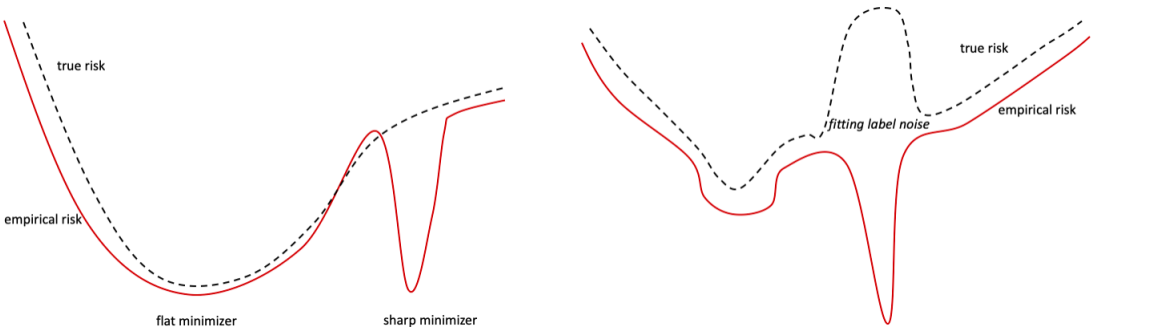
There are two distinct research areas in which we will review the background work, the first is DRM and the second is the application of ML to network intrusion detection.

After a review of previous work, we will examine the methodology used in this thesis to extend these two distinct areas of research in order to develop and test novel NIDS models.

2.1 Literature Review

2.1.1 Diametrical Risk Minimization

There is a large body of research into ways to reduce generalization error of DNNs. We focus specifically on DRM and similar techniques here. DRM seeks to reduce generalization error by minimizing the maximum empirical risk, formally defined in Section 2.2.1, in a small neighborhood of parameter space (Norton and Royset 2021; Foret et al. 2020; Zheng et al. 2021). This is different from other methods to reduce generalization error as DRM perturbs only the parameter vector as opposed to the input data as in adversarial training (Madry et al. 2017), or both the input data and the parameter vector (Wu et al. 2020). Conceptually, the goal of DRM is to find a flat minimizer as opposed to a sharp minimizer. A sharp minimizer is a minimum that has comparatively high empirical risk nearby, whereas a flat minimizer has similar or just slightly higher values of empirical risk nearby, see Figure 2.1. Therefore a flat minimizer should generalize better to test set data (Keskar et al. 2016). Theoretical benefits accrued from DRM in terms of improved generalization are identified in (Norton and Royset 2021; Tsai et al. 2021).



(a) Sharp vs. flat minimizers. (b) Sharp minimizer from label noise in training set.

Figure 2.1. These two images are a representation of an empirical risk landscape (training set) and corresponding true risk landscape (test set). In subfigure (a) we see two minimizers in the empirical risk landscape. ERM would consider these two minimizers to be the same, potentially leading to poor generalization if ERM ended up converging on the sharp minimizer. In contrast, DRM recognizes that a small perturbation of the parameter vector around the sharp minimizer would result in high empirical risk. As a result, DRM would prefer the flat minimizer which should generalize better to the test set. In subfigure (b) we see a sharp minimizer that is caused by the introduction of label noise in the training set. In this case DRM could converge to a flat local minima (up and to the left) that actually performs slightly worse in terms of training loss due to its larger empirical risk, but again generalizes much better to the test set due to lower diametrical risk. Source: Norton and Royset (2021).

In 2019 and 2020, several groups of researchers independently developed DRM with practical algorithms and supporting theory (Norton and Royset 2021; Foret et al. 2020; Zheng et al. 2021). Although developed under different names, the efforts have the same goal and overarching approach: find flat as opposed to sharp minimizers in the empirical risk landscape by solving a min-max optimization problem.

Norton and Royset (2021) discusses rates of convergence of DRM to a solution, develops a practical implementation of a stochastic gradient descent (SGD) based algorithm and then tests it in a computer vision setting. DRM is shown to outperform ERM model accuracy by as much as 40% when trained on data with corrupted labels because DRM is able to find flat minima and therefore more generalizable solutions.

Zheng et al. (2021) develops a SGD based algorithm for DRM under the name AMP and tests it in a computer vision setting. In contrast with Norton and Royset (2021), Zheng et al. (2021) trains the AMP-algorithm only on clean data, demonstrating state of the art performance compared to other regularization techniques.

Similarly, Foret et al. (2020) develops a SGD based algorithm for DRM under the name SAM and tests performance again in the computer vision setting. The SAM-algorithm shows novel state-of-the-art performance across several datasets, while also showing comparable robustness to label noise compared to other regularization techniques. A visual comparison of solutions found by SGD (solving the ERM problem) and the SAM-algorithm (solving the DRM problem) is contained in Figure 2.2.

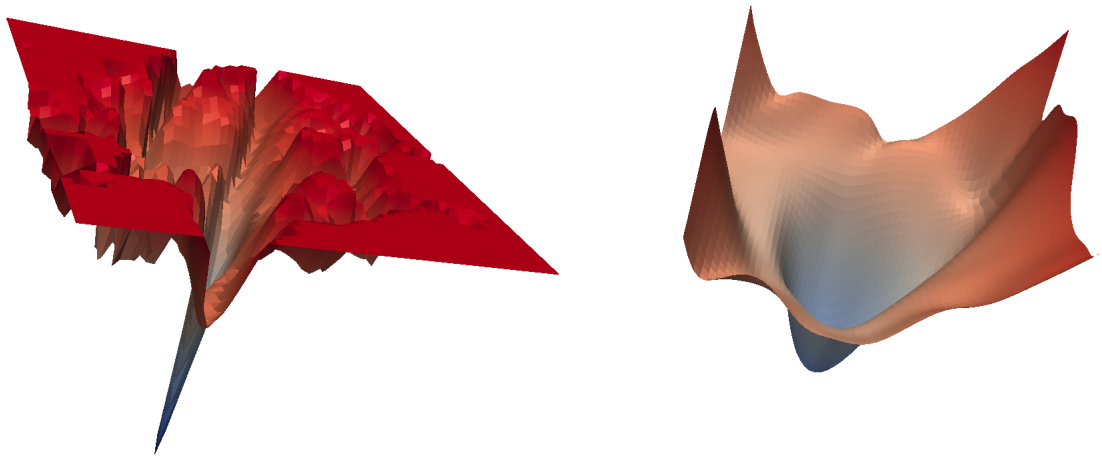


Figure 2.2. Illustration of the loss landscape defined by ResNet. It can vary greatly with many sharp minimizers (left) or appear rather smooth with flat minimizers (right). SGD tends to obtain minimizers of the kind illustrated on the left, while DRM lands at minimizers of the kind seen on the right. Source: Foret et al. (2020).

Significant amounts of research in the past few years have utilized and extended the work of Foret et al. (2020). Some research has focused on testing performance against ERM in a variety of settings (Bahri et al. 2021; Qu et al. 2022). Other research has delved into how to improve the implementation speed (Du et al. 2021; Liu et al. 2022) or performance (Kwon

et al. 2021; Kim et al. 2022; Zhuang et al. 2022) of the SAM-algorithm.

Finally, research has sought to analyze the generalization properties of SAM to understand why it performs well in certain cases (Andriushchenko and Flammarion 2022), or how converges to a minimum (Bartlett et al. 2022). Of particular interest is (Wen et al. 2022), which clarifies the underlying mechanism by which the SAM-algorithm actually minimizes sharpness in implementation. It examines the various notions of sharpness in the risk landscape and discuss the positives and negatives of minimizing each type, providing additional analytical rigor to this area of research. For broader studies of robustification of decisions in optimization models we refer to (Lewis and Pang 2010; Men et al. 2014).

2.1.2 Machine Learning in Network Intrusion Detection

There is a significant body of research into applying ML to network intrusion detection (Vinayakumar et al. 2017; Kang and Kang 2016; Song et al. 2020; Gamage and Samarabandu 2020). However, much of this existing work leverages features derived from network flow tools as inputs to the ML models (De Lucia et al. 2021). This is problematic because the ML models are not operating off of first principals, and may require a human in the loop in order to function properly. Particularly in the case of network congestion or outages, a network operator may not have a complete view of real-time traffic moving across their network.

As a solution, De Lucia et al. (2021) propose a novel type of ML model that requires only the raw payload data contained within each packet that moves across the network. This allows the resultant models to operate on first principals as opposed to requiring additional NIDS inputs or subject matter experts. Additionally, it prevents the model from merely learning a known bad Internet Protocol (IP) address or a port that is commonly used as an attack vector, neither of which are generalizable model features. Notably for our implementation, this also means there is no requirement to capture overall network flows, only the raw payload data coming into each edge computing system.

De Lucia et al. (2021) proposes two architectures for these DNNs, a one-dimensional convolutional neural network (1dCNN) and a fully connected neural network (FcNN), and then applies them to the UNSW-NB15 cyber-attack dataset. Bierbrauer et al. (2023) refines the FcNN and then applies it and the 1dCNN to two different cyber-attack datasets, UNSW-

NB15 as well as CICIDS2017, first comparing model performance then extending to a transfer learning scenario.

This recent work on FcNN and 1dCNN architectures shows great promise, with test set accuracy rates over 98% in many cases. However these models are trained on clean data, and are not tested on out-of-distribution data, making it more difficult to gauge real world performance of such a model. Recent research into the effects of poisoning attacks against ML based NIDS models found that most ML models tested showed minimal performance degradation (Talty et al. 2021), however DNNs were not considered in this work.

2.2 Methodology

DRM amounts to solving to solving a min-max optimization problem. Abstractly, we represent a neural network (NN) as a mapping $f_{\mathbf{w}}$, which takes as input a vector $\mathbf{x}$ and produces and output vector $\mathbf{y} = f_{\mathbf{w}}(\mathbf{x})$. It is parameterized by a vector $\mathbf{w}$, representing the weights of the NN. Given some loss function l, which has a positive value whenever a prediction deviates from the actual label, ERM aims to solve

$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} l(f_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$$

where $\{\mathbf{x}_i, \mathbf{y}_i, i = 1, ..., n\}$ is the training data.

In contrast, DRM shifts the focus to

$$\underset{\mathbf{w}}{\text{minimize}} \ \max_{||\mathbf{v}|| \leq \gamma} \ \frac{1}{n}\sum_{i=1}^{n} l(f_{\mathbf{w}+\mathbf{v}}(\mathbf{x}_i), \mathbf{y}_i) \tag{2.1}$$

where $\mathbf{v}$ represents a parameter perturbation bounded by a neighborhood of size γ in the norm $||\cdot||$. Thus, solving the DRM problem represents minimizing by choice of $\mathbf{w}$ the maximum empirical risk over the training data in a small γ neighborhood around the parameter vector $\mathbf{w}$.

2.2.1 Diametrical Risk Minimization Implementation

We rely on the solution approach to DRM developed in Foret et al. (2020). It approximately solves the inner maximization in (2.1) by taking a gradient ascent step (see, e.g., (Royset and Wets 2021, Section 2.D)) in the empirical risk landscape, formally defined as the graph of the function $\mathbf{w} \mapsto \frac{1}{n}\sum_{i=1}^{n} l(f_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$, and adjusting the step size such that the neighborhood constraint $||\mathbf{v}|| \leq \gamma$ is satisfied. As in Foret et al. (2020), we rely on the 2-norm to defined the neighborhood.

As in common stochastic gradient descent methods and related approaches (see, e.g., (Royset and Wets 2021, Section 3.G)), we treat the data in batches. Let $L_{\mathcal{B}}(\mathbf{w}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} l(f_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$, where $|\mathcal{B}|$ is the cardinality of $\mathcal{B}$ and thus the batch size. In detail, the algorithm of Foret et al. (2020) takes the following form.

Algorithm 1 SAM-Algorithm

Input: Training set $\{\mathbf{x}_i, \mathbf{y}_i, i = 1, ..., n\}$, batch size b, step size $\eta > 0$, neighborhood size $\gamma > 0$

Output: Trained model

Initialize model weights $\mathbf{w}_0$, set $t = 0$

while *not converged* **do**

Sample batch $\mathcal{B}$ of size b

Compute gradient $\nabla L_{\mathcal{B}}(\mathbf{w}_t)$ of batch's training loss

Compute perturbation $\mathbf{v}_t = \gamma \nabla L_{\mathcal{B}}(\mathbf{w}_t)/||\nabla L_{\mathcal{B}}(\mathbf{w}_t)||_2$

Compute gradient at perturbed point $\mathbf{g}_t = \nabla L_{\mathcal{B}}(\mathbf{w}_t + \mathbf{v}_t)$

Update weights: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$

$t = t + 1$

end while

return Model weights: $\mathbf{w}_t$

2.2.2 Data Description

We consider two datasets for this work, UNSW-NB15 (Moustafa and Slay 2015) and CICIDS2017 (Sharafaldin et al. 2018). Both are publicly available datasets that include the packet capture (PCAP) data of benign and malicious network traffic. Our subset of the

UNSW-NB15 includes nine different attack types, while our CICIDS2017 subset includes 14 different types of attacks.

We process the raw PCAP data for both datasets utilizing the Payload-Byte tool (Farrukh et al. 2022). This tool extracts payload data from raw PCAP files, labels it appropriately, and drops or pads each payload to be exactly 1500 bytes. Each packet can then be input into the DNN models as a 1x1500 vector, with each byte in the packet corresponding to one feature. A simple representation of the network packets being processed is shown in Figure 2.3.

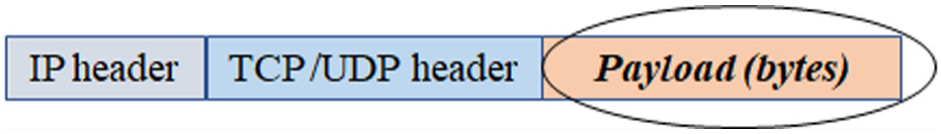


Figure 2.3. A simple example of a network packet, of which the header info is removed and just the payload data retained as the DNN input. Source: Bierbrauer et al. (2023).

We split the data 80/20 into a training and testing set, and manipulate the training set in a couple of different ways in order to test model robustness when training with ERM and DRM.

We manipulate the training set by introducing random label noise, in which a randomly selected portion of the training packets had their label flipped from "benign" to "malicious" or "malicious" to "benign." This is done to simulate the fact that in real-world applications DNN-based NIDS will not be trained on perfect, laboratory clean data. Real-world data is likely to have some mislabeling present, whether because of human error or the use of an imperfect detection and labeling tool. To be clear, this is not simulating a specific adversary poisoning attack; if an adversary has access to a NIDS model, data, etc., then there are likely more pressing issues than some label noise. This introduction of label noise is merely to explore robustness to more realistic data.

Additionally, we manipulate the training set by deliberately dropping an attack type. This is done to simulate a newly developed cyber-attack type, or zero-day exploit. In this case, the DNN-based NIDS would not have seen this exact attack before, so in our experiments it is withheld from the training set. The attack type is retained in the test set, however, so

performance can be analyzed against the novel attack type.

2.2.3 Deep Neural Network Model Architectures

We utilize two DNN model architectures based off of the work of Bierbrauer et al. (2023) and implemented in TensorFlow. Depicted in Figure 2.4, the first architecture is a FcNN, which considers each byte of the input data as a feature. The FcNN consists of 10 dense layers, using a rectified linear unit (ReLU) activation and a glorot uniform kernal initializer. A sigmoid activation function is used in the final layer to create the output. The FcNN is trained using a binary cross-entropy loss function and consists of 1,546,747 trainable parameters. The ERM model implementations utilize the Adam optimizer.

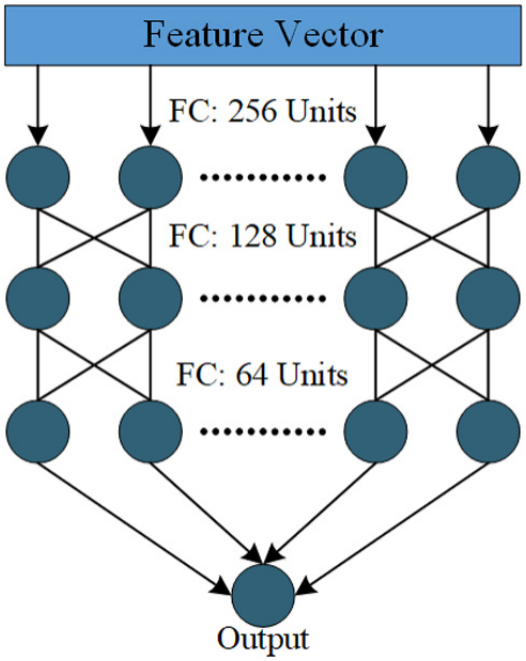


Figure 2.4. This is the architecture of a representative FcNN used for a binary classification task. This simplified FcNN consists of four dense layers with a single output of 0 or 1. Our FcNN architecture is a scaled-up version of the one shown here. Source: Bierbrauer et al. (2023).

The second DNN model architecture, depicted in Figure 2.5, is a 1dCNN consisting of two convolution layers, two dense layers and a dropout layer using a ReLU activation and

a glorot uniform kernal initializer. A sigmoid activation function is used in the final layer to create the output. The 1dCNN is trained using a binary cross-entropy loss function and consists of 4,791,857 trainable parameters. The ERM model implementations utilize the Adam optimizer. The intuition behind using a 1dCNN in this context is that there is a sort of syntax contained in a packet payload that can be beneficial in determining if the packet is malicious or benign. Using convolutional layers captures that syntax in a way that a FcNN cannot.

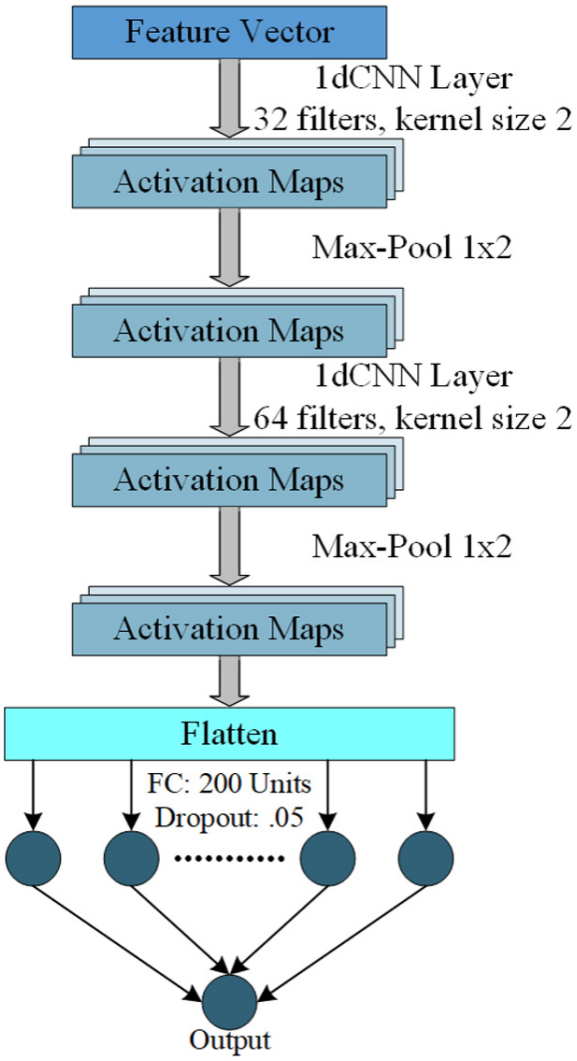


Figure 2.5. The architecture of the 1dCNN utilized in this work. Source: Bierbrauer et al. (2023).

2.2.4 Hyperparameter Tuning

We begin by exploring the optimal neighborhood size in which to solve the inner maximization problem. We explore the effect of the γ value on model performance on a subset of the UNSW-NB15 and CICIDS2017 datasets by varying the γ parameter and training the DNNs against the various manipulations of the data discussed in Section 2.2.2. For the CICIDS2017 dataset the training subset consists of over 210k packets and for the UNSW-NB15 dataset the training subset consisted of over 190k packets. The specific breakdown of attack types are shown in Tables 2.1 and 2.2, respectively.

We compare performance in relation to the size of the γ neighborhood and selected an optimal value based on the dataset, model architecture, and type of data manipulation.

Table 2.1. Breakdown of the number of training and testing packets in the subset of the CICIDS2017 data used for hyperparameter tuning.

Attack Type	CICIDS2017	
	Train	Test
None - Benign	121,735	86,246
Distributed Denial of Service (DDoS)	35,459	25,102
Secure Shell (SSH)-Patator	19,612	13,957
Infiltration	12,176	8,655
Denial of Service (DoS) Hulk	11,987	8,450
DoS Slowloris	9,817	6,930
DoS GoldenEye	452	305
DoS Slowhttptest	285	213
Web Attack – Brute Force	86	58
Heartbleed	62	46
Web Attack – XSS	35	29
PortScan	13	7
File Transfer Protocol (FTP)-Patator	3	2
Total Benign	121,735	86,246
Total Malicious	89,987	63,754

Table 2.2. Breakdown of the number of training and testing packets in the subset of the UNSW-NB15 data used for hyperparameter tuning.

Attack Type	UNSW-NB15	
	Train	Test
None - Benign	108,353	79,835
Exploits	69,211	52,858
Fuzzers	11,177	3,237
DoS	6,473	11,224
Generic	1,422	2,373
Reconnaissance	535	348
Shellcode	60	24
Backdoor	23	27
Worms	14	22
Analysis	8	52
Total Benign	108,353	79,835
Total Malicious	88,923	70,165

2.2.5 Full-Scale Runs

Utilizing our optimal γ values, we train each of the DNN model architectures on a different subset of the two datasets, and compare performance of ERM and DRM models against the various data manipulations discussed in Section 2.2.2. For the CICIDS2017 dataset the training subset consists of over 1.1 million packets and for the UNSW-NB15 dataset the training subset consisted of over 63k packets. The specific breakdown of attack types is shown in Tables 2.3 and 2.4, respectively.

Table 2.3. Breakdown of the number of training and testing packets in the subset of the CICIDS2017 data used for final results.

Attack Type	CICIDS2017	
	Train	Test
None - Benign	289,686	72,422
DoS Hulk	200,000	50,000
DDoS	193,124	48,281
DoS GoldenEye	102,497	25,625
DoS Slowloris	96,878	24,219
Infiltration	92,006	23,001
DoS Slowhttptest	64,434	16,108
SSH-Patator	38,532	9,633
FTP-Patator	25,474	6,369
Heartbleed	10,789	2,697
Web Attack – Brute Force	9,403	2,351
Web Attack – XSS	2,673	668
Bot	2,034	509
PortScan	664	166
Web Attack – Sql Injection	10	2
Total Benign	289,686	72,422
Total Malicious	838,518	209,629

Table 2.4. Breakdown of the number of training and testing packets in the subset of the UNSW-NB15 data used for final results.

Attack Type	UNSW-NB15	
	Train	Test
None - Benign	16,800	4,200
Generic	14,064	3,516
Exploits	11,193	2,799
Fuzzers	10,178	2,544
Reconnaissance	6,050	1,512
DoS	2,718	679
Backdoor	991	248
Analysis	966	242
Shellcode	870	218
Worms	74	19
Total Benign	16,800	4,200
Total Malicious	47,104	11,777

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Computational Experimentation

In this chapter, we focus on tuning the hyperparameter γ, as discussed in Section 2.2.4. After recommending γ values, we report some selected model performance on the tuning subset of data in order to get some initial idea as to how our DRM implementation is performing.

3.1 The γ Neighborhood

Foret et al. (2020) suggests that a neighborhood size of $\gamma = 0.05$ performs well over a range of computer vision tasks. However, since network intrusion detection is a different application of DNNs than computer vision, we felt it was best to explore the effect of γ in our NIDS setting. Additionally, we can get some initial comparisons between ERM and DRM.

Given two datasets, two DNN model architectures, numerous training data manipulations, and numerous γ values to try, there are a large number of DNNs to train. We train 468 models on the data subsets shown in Tables 2.1 and 2.2, testing γ values between 0.01 and 0.5 and summarize our findings after a brief note on evaluating model performance.

3.1.1 Evaluating Model Performance

When evaluating model performance, we present a number of metrics in order to quantify performance. We utilize: accuracy, defined as the percentage of correctly classified packets with a classification threshold of 0.5. True Positive Rate (TPR), or the percentage of malicious packets that are classified as malicious. True Negative Rate (TNR), or the percentage of benign packets that are classified as benign. False Positive Rate (FPR), or the percentage of benign packets that are classified as malicious and Positive Predictive Value (PPV), the number of true malicious predictions divided by the total number of malicious predictions. We also present area under the curve (AUC), a representation of the trade-off between TPR and the FPR as we vary the classification threshold from $[0, 1]$.

For models trained with a withheld attack type, we also present the performance of the

model specifically against the novel attack type. This allows us to see the percentage of the novel attack type that was correctly classified as malicious by the model. For these models, we consider the performance on the novel attack type to be the most important metric. We make the assumption that previously recognized attack vectors are known to the operator and therefore patched against. The previously unseen attack type or zero-day exploit is therefore what can actually impact our combat systems. So while it is important to accurately classify benign and known malicious traffic to recognize if our systems are under attack, it is more important to recognize if our systems are the target of an attack that could compromise them.

3.1.2 Hyperparameter γ Recommendations

By Dataset

In some cases, the CICIDS2017 data seems to perform well with a slightly higher γ value when compared to UNSW-NB15. The effect is not large, but for a given data manipulation and model architecture, we find that a CICIDS2017 DNN should utilize an equal or larger γ value than a UNSW-NB15 DNN.

By Architecture

Against every data manipulation type and dataset, the 1dCNNs seems to perform well with a higher γ value than its equivalent FcNN. We do not explore the reasons in this work, but it seems the empirical risk landscape of the 1dCNN favors larger γ values.

By Data Manipulation

When training with clean data we recommend a γ value of 0.025 for a FcNN trained on either dataset. For a 1dCNN, we recommend a γ value of 0.1 for either dataset.

When training on randomly mislabeled data, we recommend a γ value of 0.01 for a FcNN trained on either dataset. For a 1dCNN, we recommend a γ value of 0.025 for either dataset. These are smaller than the γ values recommended when training with clean data. This is due to a phenomenon that as the percentage of label noise in the training set increases and as the γ value increases, we begin to see cases where the model never moves off a naive solution of merely classifying every packet as "benign". For example, using a γ value of

0.1 or higher often results in as many as 10 failed attempts to change the random seed and output any model that predicts with any more accuracy than the naive "benign" prediction.

Against specific withheld attack types we do not encounter the same phenomena we do when training with randomly mislabeled data. In these cases we find that an equal or higher value of γ compared to our clean data testing seems to help the model better generalize to the previously unseen attack. Specifically, we recommend the following: For a FcNN trained on CICIDS2017, use γ value of 0.05 or 0.025 if trained on UNSW-NB15. For a 1dCNN trained on CICIDS2017, use γ value of 0.15 or 0.1 if trained on UNSW-NB15.

3.2 Initial Performance Comparisons

After conducting hyperparameter tuning for the γ value, we compare the performance of models trained on clean data, mislabeled data, and several withheld attack types to get a sense of the difference between ERM and DRM. Performance is summarized in Tables 3.1-3.3.

Table 3.1. Preliminary results from models trained on clean data. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold.

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
Accuracy	**99.42** / 99.41	**99.74** / 99.70	96.09 / **96.19**	96.86 / **97.29**
AUC	99.88 / **99.94**	99.96 / **99.97**	97.57 / **97.65**	99.34 / **99.43**
TPR	99.31 / **99.48**	**99.77** / 99.63	**97.84** / 97.64	95.79 / **96.84**
TNR	**99.50** / 99.35	99.71 / **99.75**	94.55 / **94.91**	**97.80** / 97.68
FPR	**0.50** / 0.65	0.29 / **0.25**	5.45 / **5.09**	**2.20** / 2.32
PPV	**99.33** / 99.13	99.61 / **99.66**	94.27 / **94.65**	**97.46** / 97.36

On clean data, ERM and DRM seem to perform similarly. Performance varies across the

various metrics by less than 0.5% with both types of minimization performing better in some but not all cases.

Table 3.2. Preliminary results from models trained with 10% random label noise. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold.

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
Accuracy	**99.07** / 98.98	**99.27** / 99.21	94.79 / **95.50**	96.82 / **96.89**
AUC	**99.83** / 99.78	99.87 / 99.87	96.19 / **96.93**	98.90 / **99.11**
TPR	**98.80** / 98.56	**99.23** / 99.09	97.62 / **97.63**	95.27 / **95.49**
TNR	99.26 / **99.29**	99.30 / 99.30	92.31 / **93.63**	**98.18** / 98.11
FPR	0.74 / **0.71**	0.70 / 0.70	7.69 / **6.37**	**1.82** / 1.89
PPV	99.01 / **99.04**	99.06 / 99.06	92.06 / **93.31**	**97.87** / 97.81

On 10% randomly mislabeled data, ERM and DRM seem to perform similarly. The exception is on the UNSW-NB15 dataset when using a FcNN; in this case DRM outperforms ERM by a small (< 2%) margin over all tracked performance metrics.

Table 3.3. Preliminary results from models trained with an attack withheld. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold. In the CICIDS2017 dataset we withhold the "Infiltration" attack and in the UNSW-NB15 dataset we withhold the "Exploits" attack.

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
Accuracy	94.54 / **96.15**	95.95 / **96.94**	94.83 / **95.53**	95.76 / **96.16**
AUC	96.52 / **99.16**	99.33 / **99.62**	96.85 / **97.28**	98.95 / **99.06**
TPR	87.75 / **91.73**	90.76 / **93.88**	94.36 / **95.62**	92.70 / **93.25**
TNR	**99.56** / 99.41	**99.78** / 99.20	95.24 / **95.45**	98.45 / **98.72**
FPR	**0.44** / 0.59	**0.22** / 0.80	4.76 / **4.55**	1.55 / **1.28**
PPV	**99.35** / 99.14	**99.68** / 98.89	94.89 / **95.15**	98.13 / **98.46**
New Attack Performance	34.91 / **42.02**	33.16 / **58.08**	94.58 / **95.92**	93.56 / **93.86**

When trained with a specific attack type withheld, DRM seems to outperform ERM more often than not. However the thing to focus on is what we consider to be the most important metric as discussed in Section 3.1.1, performance against the previously unseen attack type. On this metric DRM outperformed ERM in all four cases explored, and notably in the case of a 1dCNN trained on the CICIDS2017 dataset, DRM catches nearly twice as many malicious packets as ERM.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Model Performance and Operational Considerations

In this chapter, we begin by presenting the performance of our DNNs. We utilize the γ values recommended in Section 3.1.2. However, to ensure we are presenting objective results we train and test these networks on a different data subset than was used for tuning. After presenting results, we discuss how DRM is generalizing by examining not just the test set performance but also the training process. We then discuss the computational tractability of training with DRM as well as actually deploying our models on edge computing resources.

4.1 Deep Neural Network Results

Using our recommended values for the hyperparameter γ, we now obtain final model performance results on a different subset of data. We train 140 models on the data subsets shown in Tables 2.3 and 2.4 and summarize our findings in Tables 4.1-4.3.

Table 4.1. Results from models trained with increasing amounts of random label noise. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold.

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
		Clean Data		
Accuracy	99.74 / **99.82**	99.91 / 99.91	**98.20** / 97.83	**98.72** / 98.57
AUC	99.98 / 99.98	99.99 / 99.99	**99.77** / 99.65	99.73 / **99.75**
TPR	**99.88** / 99.87	99.93 / 99.93	**98.51** / 97.53	**98.77** / 98.63
TNR	99.33 / **99.66**	99.83 / **99.86**	97.33 / **98.69**	**98.57** / 98.38
FPR	0.67 / **0.34**	0.17 / **0.14**	2.67 / **1.31**	**1.43** / 1.62
PPV	99.77 / **99.88**	99.94 / **99.95**	99.04 / **99.52**	**99.49** / 99.42
		10% Label Noise		
Accuracy	99.74 / **99.79**	99.91 / **99.92**	98.20 / **98.47**	**98.72** / 98.64

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
AUC	99.98 / 99.98	99.99 / 99.99	99.77 / **99.81**	99.73 / **99.74**
TPR	99.88 / 99.88	99.93 / **99.94**	98.51 / **98.80**	**98.77** / 98.69
TNR	99.33 / **99.52**	99.83 / **99.87**	97.33 / **97.52**	**98.57** / 98.50
FPR	0.67 / **0.48**	0.17 / **0.13**	2.67 / **2.48**	**1.43** / 1.50
PPV	99.77 / **99.83**	99.94 / **99.95**	99.04 / **99.11**	**99.49** / 99.46
20% Label Noise				
Accuracy	**99.62** / 99.57	98.63 / **99.60**	98.05 / **98.08**	97.93 / **98.10**
AUC	**99.87** / 99.64	97.83 / **99.73**	99.42 / **99.60**	**99.20** / 99.18
TPR	**99.85** / 99.84	99.84 / 99.84	97.97 / **98.04**	98.06 / **98.32**
TNR	**98.98** / 98.77	95.13 / **98.90**	**98.26** / 98.21	**97.57** / 97.50
FPR	**1.02** / 1.23	4.87 / **1.10**	**1.74** / 1.79	**2.43** / 2.50
PPV	**99.65** / 99.58	98.34 / **99.62**	**99.37** / 99.35	**99.12** / 99.10
30% Label Noise				
Accuracy	**99.58** / 99.51	96.51 / **98.06**	97.07 / **97.62**	94.23 / **96.03**
AUC	**99.81** / 99.54	93.02 / **97.07**	**99.05** / 98.76	96.55 / **97.97**
TPR	99.74 / **99.82**	99.19 / **99.49**	97.42 / **98.20**	93.62 / **96.06**
TNR	**99.11** / 98.62	88.78 / **93.95**	**96.10** / 96.00	95.93 / 95.93
FPR	**0.89** / 1.38	11.22 / **6.05**	**3.90** / 4.00	4.07 / 4.07
PPV	**99.69** / 99.53	96.24 / **97.94**	**98.59** / 98.57	98.47 / **98.51**
40% Label Noise				
Accuracy	**99.32** / 99.29	92.67 / **96.01**	92.28 / **92.95**	87.97 / **91.53**
AUC	**99.61** / 99.14	87.76 / **97.62**	**93.74** / 93.40	91.42 / **94.74**
TPR	99.54 / **99.74**	95.75 / **99.28**	**98.20** / 96.48	87.93 / **92.63**
TNR	**98.70** / 98.00	83.76 / **86.56**	75.67 / **83.02**	88.10 / **88.45**
FPR	**1.30** / 2.00	16.24 / **13.44**	24.33 / **16.98**	11.90 / **11.55**
PPV	**99.55** / 99.31	94.47 / **95.53**	91.88 / **94.10**	95.39 / **95.74**
50% Label Noise[1]				

[1]At 50% label noise, there is significant instability in the solutions. On FcNNs the CICIDS2017 DRM model and the UNSW-NB15 ERM models both failed to converge to anything better than a naive solution of classifying everything as "malicious". This results in several metrics either being 0% or 100% correct. As

Metric	CICIDS2017		UNSW-NB15	
	FcNN	1dCNN	FcNN	1dCNN
	ERM / DRM	ERM / DRM	ERM / DRM	ERM / DRM
Accuracy	98.45 / 74.32	76.81 / **88.53**	73.71 / 47.59	**79.08** / 74.41
AUC	98.51 / 64.53	70.57 / **76.38**	86.64 / 63.73	**75.35** / 70.00
TPR	99.63 / 100.00	79.20 / **94.23**	100.00 / 29.73	**85.69** / 81.29
TNR	95.04 / 0.00	69.90 / **72.03**	0.00 / 97.69	**60.55** / 55.14
FPR	4.96 / 100.00	30.10 / **27.97**	100.00 / 2.31	**39.45** / 44.86
PPV	98.31 / 74.32	88.39 / **90.70**	73.71 / 97.30	**85.90** / 83.56

On clean training data, DRM shows equal or better accuracy compared to ERM on the CICIDS2017 data. On the UNSW-NB15 data, ERM is more accurate for both DNN types. On training data with random label noise, models utilizing DRM have higher accuracy in 13 of the 20 cases examined. ERM models have higher accuracy in 5 of the 20 cases.

In contrast to our initial results discussed in Section 3.2, we see that DRM seems to perform better on the CICIDS2017 data as opposed to the UNSW-NB15 data. We also note that while DRM clearly outperforms ERM on the CICIDS2017 1dCNN, in the case of UNSW-NB15 DRM actually performs better on the FcNN. This shows that there is not necessarily a dataset or DNN architecture that DRM works better or worse with. A particular combination of data and architecture may or may not be a good fit for DRM, so any new combination should be thoroughly tested prior to deployment.

such, we do not highlight a better performing metric for these two cases because it is disingenuous to say for example that a TPR of 100% is "better" when it stems from a naive classification.

Table 4.2. Results from models trained on the CICIDS2017 data with individual attacks withheld. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold.

Metric	CICIDS2017	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
	Withheld DoS Hulk	
Accuracy	**99.78** / 99.75	**99.92** / 99.90
AUC	99.98 / 99.98	99.99 / 99.99
TPR	99.79 / **99.87**	**99.93** / 99.89
TNR	**99.75** / 99.42	99.90 / **99.91**
FPR	**0.25** / 0.58	0.10 / **0.09**
PPV	**99.91** / 99.80	99.97 / 99.97
New Attack Performance	99.92 / **99.99**	100.00 / 100.00
	Withheld DDoS	
Accuracy	**99.82** / 99.80	**99.93** / 99.89
AUC	99.98 / 99.98	**99.99** / 99.98
TPR	99.85 / **99.88**	**99.94** / 99.89
TNR	**99.72** / 99.57	**99.90** / 99.87
FPR	**0.28** / 0.43	**0.10** / 0.13
PPV	**99.90** / 99.85	**99.97** / 99.95
New Attack Performance	99.99 / 99.99	99.99 / 99.99
	Withheld DoS GoldenEye	
Accuracy	**99.82** / 99.75	**99.93** / 99.89
AUC	99.98 / 99.98	99.99 / 99.99
TPR	99.87 / **99.88**	**99.94** / 99.89
TNR	**99.67** / 99.38	**99.90** / 99.88
FPR	**0.33** / 0.62	**0.10** / 0.12
PPV	**99.89** / 99.79	**99.97** / 99.96
New Attack Performance	99.97 / 99.97	99.98 / **100.00**
	Withheld DoS Slowloris	

Metric	CICIDS2017	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
Accuracy	**99.79** / 99.73	**99.93** / 99.89
AUC	99.99 / 99.99	99.99 / 99.99
TPR	**99.88** / 99.84	**99.94** / 99.89
TNR	**99.53** / 99.41	**99.90** / 99.87
FPR	**0.47** / 0.59	**0.10** / 0.13
PPV	**99.84** / 99.79	99.96 / 99.96
New Attack Performance	**100.00** / 99.99	100.00 / 100.00
Withheld Infiltration		
Accuracy	95.80 / **96.17**	**92.18** / 91.93
AUC	**99.16** / 98.80	98.98 / **99.04**
TPR	94.54 / **95.12**	**89.51** / 89.17
TNR	**99.44** / 99.18	99.90 / **99.91**
FPR	**0.56** / 0.82	0.10 / **0.09**
PPV	**99.80** / 99.70	99.96 / 99.96
New Attack Performance	51.42 / **56.64**	**4.97** / 2.23
Withheld DoS Slowhttptest		
Accuracy	**99.83** / 99.72	**99.93** / 99.79
AUC	**99.99** / 99.98	99.99 / 99.99
TPR	99.87 / 99.87	**99.93** / 99.76
TNR	**99.72** / 99.28	**99.90** / 99.89
FPR	**0.28** / 0.72	**0.10** / 0.11
PPV	**99.90** / 99.75	**99.97** / 99.96
New Attack Performance	99.98 / **99.99**	99.99 / 99.99
Withheld SSH-Patator		
Accuracy	**96.72** / 96.66	96.76 / **96.94**
AUC	**98.96** / 98.78	**98.99** / 98.79
TPR	**95.71** / 95.65	95.67 / **95.92**
TNR	**99.65** / 99.57	**99.92** / 99.90
FPR	**0.35** / 0.43	**0.08** / 0.10

Metric	CICIDS2017	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
PPV	**99.87** / 99.85	**99.97** / 99.96
New Attack Performance	**9.20** / 8.45	7.03 / **13.51**
Withheld FTP-Patator		
Accuracy	**98.31** / 98.22	97.70 / **97.87**
AUC	99.74 / **99.92**	**99.83** / 99.71
TPR	97.81 / **97.94**	96.94 / **97.16**
TNR	**99.76** / 99.04	99.90 / **99.92**
FPR	**0.24** / 0.96	0.10 / **0.08**
PPV	**99.91** / 99.66	99.96 / **99.97**
New Attack Performance	32.47 / **35.99**	1.74 / **14.65**
Withheld Heartbleed		
Accuracy	98.81 / **98.85**	**98.97** / 98.95
AUC	99.64 / **99.68**	99.00 / **99.49**
TPR	98.62 / **98.65**	**98.65** / 98.63
TNR	99.35 / **99.41**	99.89 / **99.90**
FPR	0.65 / **0.59**	0.11 / **0.10**
PPV	99.77 / **99.79**	99.96 / 99.96
New Attack Performance	1.00 / **3.78**	0.59 / **1.04**
Withheld Web Attack - Brute Force		
Accuracy	**99.74** / 99.52	**99.77** / 99.49
AUC	**99.98** / 99.97	99.99 / 99.99
TPR	**99.88** / 99.73	**99.72** / 99.34
TNR	**99.34** / 98.90	99.90 / 99.90
FPR	**0.66** / 1.10	0.10 / 0.10
PPV	**99.77** / 99.62	99.97 / 99.97
New Attack Performance	**100.00** / 90.17	**81.03** / 51.04
Withheld Web Attack - XSS		
Accuracy	**99.81** / 99.75	**99.93** / 99.79
AUC	99.98 / 99.98	99.99 / 99.99
TPR	**99.85** / 99.84	**99.93** / 99.76

Metric	CICIDS2017	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
TNR	**99.69** / 99.50	**99.92** / 99.86
FPR	**0.31** / 0.50	**0.08** / 0.14
PPV	**99.89** / 99.83	**99.97** / 99.95
New Attack Performance	98.80 / **99.25**	99.40 / **99.70**
	Withheld Bot	
Accuracy	99.68 / **99.69**	**99.80** / 99.79
AUC	**99.96** / 99.94	99.92 / **99.93**
TPR	99.72 / **99.74**	**99.77** / 99.75
TNR	**99.58** / 99.55	99.90 / 99.90
FPR	**0.42** / 0.45	0.10 / 0.10
PPV	**99.86** / 99.84	**99.97** / 99.96
New Attack Performance	**1.96** / 0.98	**7.47** / 2.16
	Withheld PortScan	
Accuracy	**99.77** / 99.71	**99.92** / 99.89
AUC	**99.98** / 99.97	99.99 / 99.99
TPR	99.87 / **99.89**	**99.91** / 99.89
TNR	**99.46** / 99.20	**99.94** / 99.91
FPR	**0.54** / 0.80	**0.06** / 0.09
PPV	**99.81** / 99.73	**99.98** / 99.97
New Attack Performance	91.57 / **96.99**	91.57 / **94.58**
Withheld Web Attack - Structured Query Language (SQL) Injection		
Accuracy	**99.81** / 99.71	**99.93** / 99.80
AUC	**99.99** / 99.98	99.99 / 99.99
TPR	99.81 / **99.88**	**99.93** / 99.76
TNR	**99.81** / 99.22	**99.91** / 99.89
FPR	**0.19** / 0.78	**0.09** / 0.11
PPV	**99.93** / 99.73	**99.97** / 99.96
New Attack Performance	50.00 / **100.00**	0.00 / 0.00

When an attack type is withheld from the training set but retained in the test set, we are primarily concerned about the performance of the resulting model against that previously unseen attack. In the case of the CICIDS2017 data, we find that DRM outperforms ERM in 14 of the 28 cases examined. ERM outperforms DRM in 7 of the cases. The remaining 7 cases are ties. Importantly, the better performance of DRM trained models against previously unseen attack types is not just confined to one architecture; DRM outperforms ERM on both architectures examined.

Table 4.3. Results from models trained on the UNSW-NB15 data with individual attacks withheld. All values are given in percentages with the better performing metric for each ERM / DRM comparison in bold.

Metric	UNSW-NB15	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
	Withheld Generic	
Accuracy	**98.44** / 98.08	**98.76** / 98.55
AUC	**99.74** / 99.71	99.76 / 99.76
TPR	**98.23** / 97.73	**98.71** / 98.46
TNR	99.02 / **99.05**	**98.90** / 98.79
FPR	0.98 / **0.95**	**1.10** / 1.21
PPV	99.65 / 99.65	**99.61**/ 99.56
New Attack Performance	98.92 / **98.98**	99.09 / **99.12**
	Withheld Exploits	
Accuracy	98.13 / **98.34**	**98.42** / 98.25
AUC	99.69 / **99.70**	**99.72** / 99.70
TPR	98.12 / **98.24**	98.08 / **98.36**
TNR	98.14 / **98.60**	**99.36** / 97.95
FPR	1.86 / **1.40**	**0.64** / 2.05
PPV	99.33 / **99.49**	**99.77** / 99.26
New Attack Performance	94.68 / **95.07**	94.93 / **95.61**
	Withheld Fuzzers	
Accuracy	94.77 / **96.61**	**97.03** / 97.01

Metric	UNSW-NB15	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
AUC	98.63 / **99.42**	99.55 / **99.58**
TPR	93.39 / **95.86**	**97.64** / 96.31
TNR	98.67 / **98.74**	95.33 / **98.95**
FPR	1.33 / **1.26**	4.67 / **1.05**
PPV	99.49 / **99.53**	98.32 / **99.61**
New Attack Performance	74.76 / **86.52**	**91.16** / 89.78
Withheld Reconnaissance		
Accuracy	**98.30** / 97.62	**98.39** / 98.08
AUC	**99.75** / 99.70	**99.72** / 99.71
TPR	**98.90** / 97.10	**99.24** / 97.91
TNR	96.64 / **99.07**	96.02 / **98.57**
FPR	3.36 / **0.93**	3.98 / **1.43**
PPV	98.80 / **99.66**	98.59 / **99.48**
New Attack Performance	**95.57** / 94.51	**98.02** / 97.09
Withheld DoS		
Accuracy	**98.65** / 95.62	**98.59** / 98.43
AUC	**99.78** / 99.39	**99.74** / 99.72
TPR	98.64 / **98.91**	**98.62** / 98.36
TNR	**98.67** / 86.38	98.48 / **98.62**
FPR	**1.33** / 13.62	1.52 / **1.38**
PPV	**99.52** / 95.32	99.45 / **99.50**
New Attack Performance	97.64 / **98.09**	**96.47** / 94.26
Withheld Backdoor		
Accuracy	98.60 / 98.60	**98.81** / 98.39
AUC	99.76 / **99.82**	**99.75** / 99.72
TPR	98.79 / **98.96**	**99.02** / 98.46
TNR	**98.05** / 97.62	**98.24** / 98.19
FPR	**1.95** / 2.38	**1.76** / 1.81
PPV	**99.30** / 99.15	**99.37** / 99.35

Metric	UNSW-NB15	
	FcNN	1dCNN
	ERM / DRM	ERM / DRM
New Attack Performance	95.97 / **97.58**	94.76 / **97.58**
Withheld Analysis		
Accuracy	**98.53** / 98.48	**98.77** / 98.57
AUC	99.77 / **99.81**	**99.79** / 99.75
TPR	98.64 / **98.77**	**98.61** / 98.58
TNR	**98.21** / 97.67	**99.21** / 98.55
FPR	**1.79** / 2.33	**0.79** / 1.45
PPV	**99.36** / 99.16	**99.72** / 99.48
New Attack Performance	100.00 / 100.00	100.00 / 100.00
Withheld Shellcode		
Accuracy	98.23 / **98.38**	98.43 / 98.43
AUC	99.68 / **99.77**	99.70 / **99.72**
TPR	97.88 / **98.69**	98.29 / **98.40**
TNR	**99.24** / 97.50	**98.81** / 98.52
FPR	**0.76** / 2.50	**1.19** / 1.48
PPV	**99.72** / 99.10	99.57 / **99.47**
New Attack Performance	93.58 / **94.95**	88.99 / **93.58**
Withheld Worms		
Accuracy	97.21 / **98.52**	98.52 / **98.69**
AUC	99.57 / **99.75**	99.73 / **99.74**
TPR	98.51 / **98.59**	**99.71** / 98.63
TNR	93.55 / **98.31**	95.19 / **98.86**
FPR	6.45 / **1.69**	4.81 / **1.14**
PPV	97.72 / **99.39**	98.31 / **99.59**
New Attack Performance	100.00 / 100.00	89.47 / **100.00**

In the case of the UNSW-NB15 data, we find that DRM outperforms ERM in 11 of the 18 cases examined. ERM outperforms DRM in four of the cases. The remaining three cases

are ties. Again, the better performance of DRM trained models against previously unseen attack types is observed on both model architectures examined.

4.2 Is Diametrical Risk Minimization Generalizing Better?

In order to understand where the better performance of DRM is coming from, it is useful to examine a few specific examples in greater depth. First we will examine the case of a 1dCNN trained on clean CICIDS2017 data, shown at the top of Table 4.1.

In this case, ERM and DRM performed the same on Accuracy, AUC, and TPR, while DRM performed better on TNR, FPR, and PPV. We conclude that DRM outperforms ERM and, by looking at the training process, we can see how. Since the training data is clean, we are not concerned about sharp minimizers caused specifically by label noise. What we see is that the ERM trained model converged to a solution with a training loss of 0.00095, whereas the DRM trained model converged to an objectively worse solution for the training set, with a loss of 0.00211. The more than 2x higher training loss for the DRM solution is also reflected in the accuracy on the training set, with DRM showing an accuracy of 0.9997 to ERM's 0.9998.

However when it comes to test set performance DRM outperforms ERM. Since the training set performance of DRM was objectively worse, we conclude that that the DRM trained model converged to a solution which generalized better to the test set. In a clean data setting, avoiding sharp minimizers in the risk landscape led to a more generalizable solution; exactly what we hope DRM does for us.

The second case we examine is of a 1dCNN trained on CICIDS2017 data with 30% random label noise in the training set, shown in the center of Table 4.1. In this case, the DRM trained model outperformed ERM in all six performance metrics tracked. Here we are concerned about sharp minimizers in the risk landscape that are caused by label noise, see Figure 2.1, because we know these solutions will not generalize well. We see is that the ERM trained model converged to a solution with a training loss of 0.48651, whereas the DRM trained model converged to a solution with a loss of 0.49730. Again we see that the DRM solution shows worse accuracy on the training set when compared to ERM, 0.7915 to 0.7948.

Against the properly labeled test set data, the DRM model performs better. In this case we conclude that by avoiding sharp minimizers, potentially caused by label noise in the training set, DRM again converged to a more generalizable solution.

These two cases show that implementing DRM to train our DNNs is working the way we want it to. DRM is not performing well by achieving the lowest training loss values, instead it is converging to flatter solutions which may then generalize better. DRM is not magical, but more often than not it converges to more generalizable solutions, especially when faced with tasks requiring additional model robustness like mislabeled training data or novel attack types.

4.3 Operational Considerations

Training utilizing DRM is slower than training with ERM; however, we find that the choice of DNN architecture matters more to the training time than the ERM or DRM choice does. Depending on the dataset and architecture, we observe that training with DRM takes 1.5-1.9x longer than training the same model using ERM. In contrast, depending on the dataset and choice of ERM or DRM, we observe that training a 1dCNN takes 3.3-6.2x as long as the equivalent FcNN.

Finally, we explore the feasibility of deploying a NIDS such as this on edge computing resources. In order to test this, we take our trained DNN models and utilize them to classify test set packets on a 2021 MacBook Pro containing an Apple M1 chip. This simulates an edge computing system using commercial-off-the-shelf (COTS) hardware to classify incoming packets. Training the models with ERM or DRM no longer matters for this speed test, as the only difference in the resultant model is the values of the DNN weights. We find that using a FcNN model the M1 chip can process and then classify incoming packets at a rate of 25.4 MBps. For the 1dCNN, we achieve a rate of 9.4 MBps.

This test shows that running a NIDS such as this on COTS hardware is not only tractable, but can be done at relevant speeds. For comparison, Link-16, the ubiquitous datalink found across the U.S. and allied militaries, operates in the kbps range. In this case, either of our DNN models could be used to examine every incoming Link-16 transmission in real time to help harden the networked combat system against cyber-attacks.

CHAPTER 5: Conclusion

We find that with clean data and no novel attack types, a model trained with DRM may be more accurate but it is dependent on the combination of dataset and DNN architecture. Since it is not clear if training with DRM is best in this context, we conclude that our implementation of DRM does not improve on current state-of-the-art performance in this use case. However, this use case is probably the least realistic scenario we would find in deployment of a NIDS.

Where DRM clearly outperforms ERM is in cases where additional model robustness is required. When some of the training data is mislabeled, DRM is more accurate than ERM in 13 of the 20 cases we examine, while ERM is shows greater accuracy than DRM in 5 cases. With realistic amounts ($\leq$ 20%) of mislabeling present, these models show test set accuracy rates greater than 98%.

Similarly, when tested against previously unseen attack types, DRM trained models outperform ERM trained models at correctly classifying the novel attack type as "malicious". Of the 46 cases presented, models trained with DRM show better performance compared to models trained with ERM in 25 cases and equal or better performance in 35 cases.

Importantly, when faced with mislabeled training data or novel attack types, DRM outperforms ERM across both datasets and DNN architectures. This shows that the additional robustness exhibited by DRM is not merely a fluke of a particular dataset and architecture combination. We find that while DRM might not show improvements over ERM in the lab, its additional robustness means that it will likely outperform ERM in more realistic deployment scenarios.

Further, we show that while training with DRM does take additional compute resources compared to ERM, the choice of DNN architecture is actually more impactful on total resources required. Then regardless of the choice of ERM or DRM, either of these DNN model architectures are computationally tractable on edge computing systems using COTS hardware. Therefore we conclude that training DNNs with DRM provides additional model

robustness and recommend that DRM be utilized if the compute resources can support it.

5.1 Future Work

Future work in this research area could proceed in few different directions. One potential area of work is the choice of algorithm to approximate the inner maximization problem of DRM, either to speed up the implementation or to obtain a better approximation.

In our implementation we are inherently making the assumption that the gradient of the batch's training loss represents the direction of the maximum empirical risk in the γ neighborhood. This may be a reasonable assumption for very small γ values, but clearly as the γ neighborhood grows this assumption is likely invalid. Norton and Royset (2021) make a different assumption in their algorithm; namely, that sampling random 2-norm bounded perturbations in parameter space will find a region of the γ that approximates the location of the maximum empirical risk. The DNN performance and robustness implications of these assumptions inherent to different DRM implementations could be compared using a similar methodology to what we present in this thesis.

Another area of potential future work is in testing DRM performance when classifying specific attack types as opposed to just a binary "benign" or "malicious" classification. Such a model could provide more useful insight during operational deployment, where attack trends could be monitored with additional granularity.

Finally, one could compare ERM and DRM performance when testing on unbalanced cyber-attack data, which would be more indicative of real-world performance. There may be a need to balance the training data in order for the DNNs to learn what cyber-attacks look like. However during deployment, we would expect the vast majority of network traffic to be benign. Testing on data that is more representative of what a system is likely to see in the real-world would extend this work, and help inform whether or not a DNN based NIDS is ready for operational deployment.

List of References

Andriushchenko M, Flammarion N (2022) Towards understanding sharpness-aware minimization. *International Conference on Machine Learning* (PMLR), 639–668.

Bahri D, Mobahi H, Tay Y (2021) Sharpness-aware minimization improves language model generalization. *arXiv preprint arXiv:2110.08529*.

Bartlett PL, Long PM, Bousquet O (2022) The dynamics of sharpness-aware minimization: Bouncing across ravines and drifting towards wide minima. *arXiv preprint arXiv:2210.01513*.

Bierbrauer DA, De Lucia MJ, Reddy K, Maxwell P, Bastian ND (2023) Transfer learning for raw network traffic detection. *Expert Systems with Applications* 211:118641.

De Lucia MJ, Maxwell PE, Bastian ND, Swami A, Jalaian B, Leslie N (2021) Machine learning raw network traffic detection. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, volume 11746 (SPIE), 185–194.

Du J, Yan H, Feng J, Zhou JT, Zhen L, Goh RSM, Tan VY (2021) Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*.

Farrukh YA, Khan I, Wali S, Bierbrauer D, Pavlik JA, Bastian ND (2022) Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets. *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)* (IEEE), 58–67.

Foret P, Kleiner A, Mobahi H, Neyshabur B (2020) Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Gamage S, Samarabandu J (2020) Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications* 169:102767.

GAO (2022) Dod cybersecurity: Enhanced attention needed to ensure cyber incidents are appropriately reported and shared. GAO Report GAO-23-105084, Washington, DC, USA, https://www.gao.gov/assets/gao-23-105084.pdf.

Kang MJ, Kang JW (2016) Intrusion detection system using deep neural network for in-vehicle network security. *PloS one* 11(6):e0155781.

Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP (2016) On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kim M, Li D, Hu SX, Hospedales T (2022) Fisher sam: Information geometry and sharpness aware minimisation. *International Conference on Machine Learning* (PMLR), 11148–11161.

Kwon J, Kim J, Park H, Choi IK (2021) Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. *International Conference on Machine Learning* (PMLR), 5905–5914.

Lewis AS, Pang CJ (2010) Lipschitz behavior of the robust regularization. *SIAM Journal on Control and Optimization* 48(5):3080–3104.

Liu Y, Mai S, Chen X, Hsieh CJ, You Y (2022) Towards efficient and scalable sharpness-aware minimization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12360–12370.

Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Men H, Freund RM, Nguyen NC, Saa-Seoane J, Peraire J (2014) Fabrication-adaptive optimization with an application to photonic crystal design. *Operations Research* 62(2):418–434.

Microsoft Threat Intelligence (2023) Volt typhoon targets us critical infrastructure with living-off-the-land techniques. May 24, https://www.microsoft.com/en-us/security/blog/2023/05/24/volt-typhoon-targets-us-critical-infrastructure-with-living-off-the-land-techniques/.

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

Norton MD, Royset JO (2021) Diametrical risk minimization: Theory and computations. *Machine Learning* 1–19.

Qu Z, Li X, Duan R, Liu Y, Tang B, Lu Z (2022) Generalized federated learning via sharpness aware minimization. *International Conference on Machine Learning* (PMLR), 18250–18280.

Royset JO, Wets RJB (2021) *An Optimization Primer*, 1st ed. (Springer Cham, Cham, Switzerland).

Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1:108–116.

Song HM, Woo J, Kim HK (2020) In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications* 21:100198.

Talty K, Stockdale J, Bastian ND (2021) A sensitivity analysis of poisoning and evasion attacks in network intrusion detection system machine learning models. *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)* (IEEE), 1011–1016.

Tsai YL, Hsu CY, Yu CM, Chen PY (2021) Formalizing generalization and robustness of neural networks to weight perturbations. *arXiv preprint arXiv:2103.02200*.

Vinayakumar R, Soman K, Poornachandran P (2017) Applying convolutional neural network for network intrusion detection. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (IEEE), 1222–1228.

Wen K, Ma T, Li Z (2022) How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*.

Wu D, Xia ST, Wang Y (2020) Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems* 33:2958–2969.

Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2021) Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* 64(3):107–115.

Zheng Y, Zhang R, Mao Y (2021) Regularizing neural networks via adversarial model perturbation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8156–8165.

Zhuang J, Gong B, Yuan L, Cui Y, Adam H, Dvornek N, Tatikonda S, Duncan J, Liu T (2022) Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California