Crafting Machine Learning Defenses against Adversaries

by

Won Park

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Z. Morley Mao, Chair
Assistant Professor David Fouhey
Professor Atul Prakash
Professor Clayton Scott

Won Park
wonpark@umich.edu
ORCID iD: 0000-0002-30322-0748

DEDICATION

To all those who made this journey possible

ACKNOWLEDGMENTS

"It takes a village to raise a child".

I would bet this is one of the most common quotes put in the front of a dissertation, but I believe it really is true. One of the most challenging, frustrating, emotional, and rewarding experiences in my life is coming to an end. And the journey would not have been possible for the village around me. In another universe, if it weren't for the people listed here, this dissertation might look very different (or might not even exist entirely). So, near the end of my journey, I think it is fitting that I name the people who have helped me along the way.

First, I would like to give my utmost thanks to Professor Z. Morley Mao, who took me in, provided mentorship and guidance, and provided support through a PhD journey that was most certainly non-linear.

I would also like to express my sincere gratitude and respect to members of my committee: Assistant Professor David Fouhey, Professor Atul Prakash, and Professor Clayton Scott. They are very accomplished and busy individuals - yet they have always found the time to provide feedback and assistance.

Many of these projects literally would not have happened without the collaborators I worked with along the way. So thank you to Assistant Professor Qi Alfred Chen, Nan Liu, Assistant Professor Ruoxi Jia, Si Chen, and Yi Zeng. It has been the utmost privilege working alongside all of you and I eagerly look forward to all your future works.

Many thanks to my manager at Ericsson, Dr. Wenting Sun, and the team. I had a great time there and I was so blessed to have worked with such a wonderful manager and amazing team who welcomed me from day 1 and helped me become the best I can be.

I would like to thank members of my lab, past and present (Can Carlack, Yulong Cao, Minkyoung Cho, David Hong, Shengtuo Hu, Shouwei Jin, Yikai Lin, Wenyuan Ma, Yuru Shao, Jiachen Sun, Shichang Xu, Shawn Zhao, Xumiao Zhang, Qingzhao Zhang, Ruiyang Zhu, Xiao Zhu). Working alongside you all has been quite an experience and I am sad that COVID put an end to our fun meetings. I would like to give a special warm thanks to Yulong Cao, Shenguo Hu, and Yikai Lin, who were like older brothers to me when I was

new to the program; and a shoutout to others in my cohort: Can Carlack, Jiachen Sun, and Xumiao Zhang. You are all close and I look forward to your dissertations as well!

I would also like to thank other graduate students who provided guidance and were always down to have fun conversations along the way: Noah Curran, Meixing Dong, Ayush Goel, and Ramakrishnan Sundara Raman.

Jumping back in time, I would like to thank Professor David Wagner who started me on this journey, not just in terms of research in computer science but security of machine learning. Without him, I would most likely have never considered this path. Alongside him, I would also like to thank my first ever collaborators: Michael McCoyd, Steven Chen, and Joanna Yang.

The next part of my acknowledgements goes to the fabulous people who have supported me along the way. The journey was not easy (as many of them may know), and without these people on the sideline, I probably would not have made it this far.

A warm thanks to my wonderful family who have shown nothing but love and support throughout this entire journey. They have always stood by me and cared for me. I love them deeply. I would also like to thank my grandparents and relatives in Korea who I was not able to see throughout the program. I knew they were always there and supporting me, even if it could only be through a phone call or Kakao video chat. I miss them all and hope I can visit soon.

To Iris Li, the love of my life who has stood by me through the ups and downs. There is no one else I would have rather taken this journey with. Thank you for accepting me, supporting me, and be willing to share both my pains and my joys. Your emotional support was absolutely essential to me and I hope we can continue to chase our dreams together.

I would like to thank the Thursday Dinners and Tomfoolery Squad (Caleb Belth, Eli Goldweber, Sarah Jabbour, Fahad Kamran, Meera Krishnamoorthy, Kevin Loughlin, Trevor Odelberg). Not only can a PhD journey be challenging, but it is uniquely so in a way that few may be able to understand. I am grateful for my friends who were able to support me through the challenges of a PhD. I look forward to seeing what exciting places they end up.

A big thank you to the Potluck squad (Stanley Ho, Kanak Kapoor, Jong Ha Lee, Megan Lew, Cindy Liu, In-Young Jo, Sam Pajuleras, Michelle Sou, Faith Szeto, Vaibhav Ramamoorthy, Justin Wang, Julie Xin) who remind me that despite long periods of time and even longer distances, friends can always meet up over food and good times and act like not a moment was lost.

Thanks to the high king crew (Jared Fernandez, Nikhil Ghosh, Vikram Idury, Siddharth Mehendale, Kentaro Mori, Rohan Subramaniam, Hardi Vajir, Erik Yang) who have taught me that top-tier deep, intellectual conversations and top-tier degeneracy sometimes do go hand-in-hand.

I would like to thank the Heok group (Shivaani Gandhi, Brian Giang, Darren Lim, Divya Natesan, (Dr.) Sajan Patel) for reminding me despite the fact that I left the field of chemistry long ago, the bonds of friendship never break.

Thank you to the Nerd Herd Squad (Rohit Braganza, Jay Desai, Andrew Kim, Ayush Mehra, Vaibhav Ramamoorthy) who have been my support and source of laughter since middle school.

I would also like to give my sincerest thanks to my other high school and college friends who provided help, advice, laughter, unconditional support, and kindness throughout: Ji Hyun An, Katherine Jung, Rajeev Parvathala, Aleksandra Popovic, Diana Zhou.

The journey was arduous and besides my friends and family, another thing that kept me going were my hobbies that I came to enjoy. So thank you so much to the Michigan volleyball community as well as the awesome photography community who welcomed me in with warm arms and really helped me enjoy the hobby when the times got tough.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

Machine learning systems are becoming widely adopted and ubiquitous. Not only are there a growth of products in which machine learning is at their core like autonomous vehicles, but even traditional companies in fields such as finance, telecommunications, and travel are integrating machine learning into their internal structure.

However, like any system, machine learning platforms are prone to security risks and vulnerabilities. Coupled with an ever-accelerating deployment and usage of machine learning systems, the attackers' chance of success and capability of damage increases just as rapidly. What is especially concerning is the large surface area of the machine learning pipeline that is available for attack - from training all the way to inference. With such a wide variety of attack combinations possible, there remains a need to address and explore the many types of attacks and defense that are possible in a machine learning environment.

To address this goal, in this dissertation, we explore some of the different types of security vulnerabilities and attacks that are possible with different types of machine learning systems. At the inference level, my dissertation explores the possibility of crafting adversarial examples on multimodal sensor fusion models - the kind that would be used by autonomous vehicle manufacturers. We also explore a new technique that can be used for defending against adversarial examples: adversarial fine-tuning. Our proposed methodology exceeds the state of the art in terms of robust accuracy and benign accuracy, while still taking much faster to train than traditional adversarial training.

We also explore a gap in the study of attacks during the training phase of the model (i.e.

poisoning or backdoor attacks), by exploring the frequency domain of images and how that could affect attacks and detection defenses.

Finally, through a collaboration at Ericsson, Inc., we explore how a machine learning framework can be deployed to detect anomalous data while still being cognizant of industry restrictions and metrics.

CHAPTER 1

Introduction

The world is in a new era in which machine learning (ML) systems are becoming widely pervasive. Andrew Ng famously talked about "Why AI is The New Electricity" speaking about its importance in the near future [125]. Though there are reasonable critiques to his statement, it is an unequivocal fact that machine learning systems are on the rise. The global machine learning market is expected to grow from $21.17 billion in 2022 to $209.91 billion by 2029, at a CAGR of 38.8% [43] 35% of companies report using AI in their business, and an additional 42% of respondents say they are exploring AI [69]. 46% of organizations report to be planning to implement AI in the next three years [134]. This is a trend that is not limited to just one area or field, as machine learning usage is predicted to rise across different sectors such as banking, healthcare, marketing, and retail.

This trend is no surprise as machine learning systems provide the benefit of automation of almost human-like tasks, smarter decision making algorithms, new opportunities for innovation, and enhanced quality of life. Machine learning systems are able to beat humans in a game of Go [154], Atari, [115], and Super Smash Bros Melee [42]. Several companies are toying with the idea of autonomous vehicles, and recently ChatGPT made headlines for providing almost human-like communication. Recently, an artificially generated photo fooled judges and won a photography contest [52].

Behind all these amazing tasks lies one encompassing idea. At a high-level, the goal

of most machine learning tasks is to learn a rough function (albeit possibly a complicated one) from input to output given samples of input-output pairs. Machine learning algorithms are designed to learn this function using the sample of input-output pairs - called *training data* and then apply it to new, unseen data. However, a key presumption is made here: the aforementioned algorithms are developed based on a strong assumption that the distribution of the training data is similar to the new data the model will see. This assumption, while usually valid, raises some potential concerns:

The main one we are focused on this dissertation is scenarios where malicious actors (adversaries) attack a machine learning model, either by altering the training data or altering the unseen data. In either scenario, the machine learning model faces a situation where the new examples it sees are different from the types of inputs trained on.

The focus of this dissertation is on the attacks that arise from *purposeful* data manipulation of machine learning systems - which may even be malicious in nature - and potential defenses against them.

1.1 Motivation

With such advanced systems, it is only logical that it will bring about its own risks from failures and even worse, malicious agents - bringing about its own security challenges. Furthermore, as the use of machine learning continues to rise, the security threat grows with it. This is not only from the perspective of likelihood - as more machine learning systems increases, the chance of a successful attack increases as well - but also from a magnitude perspective - as more and more critical structures utilize machine learning, the consequences of an attack magnifies.

We have already seen examples of such intentional manipulation, not just in theory but in practice. One of the earliest applications of machine learning was to detect emails as

spam - and it was shortly after that attackers were able to launch attacks do disguise emails to evade detection [38]. Other examples that show potential real-life dangers include a report that showed a few stickers could trick a Tesla autopilot [86], a journal that showed a few changes undetectable by humans can cause misdiagnoses in medical systems [41], or a paper showing how to fool DeepFake detection systems [14]

Furthermore, despite the previously mentioned explosion in machine learning systems, the 2019 interim report published by the National Security Commission on Artificial Intelligence [146] shows that the funding for defending against attacks on machine learning systems lags behind. Another publication by IARPA signaled the alarm for the increasing threat of attacks during training time of machine learning models [68]. What complicates the issue of security in machine learning is that there is no one silver bullet - no one-size-fits-all solution. For instances, attacks can occur during different phases in the machine learning model like training or evaluation. Even based on the type of machine learning model, there may arise differences and similarities in security challenges - a machine learning task focused on object detection in an autonomous vehicle may face different challenges than a machine learning model used to predict stock movements. On top of all of this, defenses must be aware of different restrictions and threat models that remain acceptable based on matters like use case, industry restrictions, and general risk tolerance.

When grappling with the problem of how to manage and search for vulnerabilities in machine learning pipelines, there are many approaches one can take. In this research, I start by examining the machine learning training pipeline and identifying the security vulnerabilities that may arise in each stage:

- **Inference Phase** A critical phase of any machine learning system is the inference phase, in which a trained model is deployed. This is also where the model is subject to unconstrained inputs that are out of its control. Though there have been thou-

sands of works since the initial discovery of attacks during the inference phase [164], several areas remain unexplored. We explore a few of these areas in this dissertation.

- **Training Phase** Machine learning models are also vulnerable during their training phase through backdoor attacks. Backdoor attacks are attacks in which adversaries deliberately manipulate some subset of the training data [53, 28], or the model's parameters [104], to make the model recognize a backdoor trigger as the desired target label(s). When the backdoor trigger is introduced during test-time, the poisoned model exhibits a particular output behavior of the adversary's choosing (e.g., a misclassification).

My work brings about novel contributions for security of machine learning models in both these areas.

Furthermore, security in machine learning is not just limited to the different stages in their pipeline, but also comes into play in different deployment environments. These security measures also do not exist in a vacuum and must fit in with the surrounding deployment architecture and strategy. They must be cognizant, for example, in the limitations of time, resources, energy, and money. Toward this, I have performed part of my research at Ericsson Inc. helping research and develop security for their machine learning models.

1.2 Research Questions

In this thesis, we address three main research questions centered on examining the security of machine learning model from different attack surface perspectives.

RQ 1: What are some current threats to models during inference time and can we build effective defenses? By the time this work was written, it has been well known

that machine learning models - particularly deep learning models are vulnerable to attacks during the inference phase with what are called *adversarial examples* [164, 48].

Since its discovery, much of the work on adversarial examples has been focused on single input machine learning models like basic image classification models or single-input object detection models. While it is well known that single input machine learning models are vulnerable to adversarial examples, it is not clear what occurs if an additional input is fed into the model.

We explore this gap in Chapter 3 and show that multimodal input models are not inherently more secure than single input models (in this case: 3D depth and image). We find that despite the use of a 3D depth input, the model is vulnerable to our purposefully crafted image-based adversarial attacks including disappearance, universal patch, and spoofing. After helping to shed light on the underlying reason, we explore some potential defenses and provide some recommendations for improved sensor fusion models.

In terms of defense, the ideal goal to protect against adversarial examples is to create models that are *certified* to be resilient against distortions in inputs. This involves developing a training methodology (and possibly a corresponding model) that provides provable guarantees that the model will not misclassify any inputs within a certain epsilon of distortion. However, progress in this line of work is slow and to date, the guarantees are still quite small to be practical in real life. Because of this, the most popular defense used is an *empirical* defense that utilizes training on both regular and adversarial examples. The downside to this methodology is that it vastly increases the training time.

In Chapter 6, we provide a new training methodology that utilizes a pre-trained model and fine-tunes the model with adversarial examples for a few epochs. Our evaluation shows that it surpasses state of the art fine-tuning methods in terms of accuracy and protection against adversarial examples, even matching the accuracies provided by regular adversarial tuning techniques. At the same time, our technique takes much less time than regular

adversarial training.

RQ 2: How can we improve defenses against attacks during training phase? In contrast to adversarial examples, which attacks the model during the *inference* phase, poisoning or backdoor attacks attack a model during its *training* phase. There have been lots of works designing more and more effective attacks and even more works trying to design defenses to mitigate said attacks. However, in our dissertation (Chapter 4), we show that many current state-of-the-art backdoor attacks exhibit severe high-frequency artifacts, which remain across different types of datasets and resolutions. We further demonstrate these high-frequency artifacts can be taken advantage of and show we can detect these triggers at a rate of 98.50% without prior knowledge of the attack details or the target model. We then propose a practical way to create smooth backdoor triggers without high-frequency artifacts and study their detectability. We show that existing defense works can benefit by incorporating these smooth triggers into their design consideration. In short, we provide an exploration into a new direction of data poisoning attacks.

RQ 3: How can we provide a defense against anomalous inputs in industry systems? Security systems do not operate in a vacuum - any defense must be cognizant of the constraints of the environment it is placed in. After all, a secure defense is useless if it is so cumbersome that it is not used.

As a case study, a modern network and telecommunication systems, hundreds of thousands of nodes are interconnected by telecommunication links to exchange information between nodes. The complexity of the system and the stringent requirements on service level agreement makes it necessary to monitor network performance intelligently and enable preventative measures to ensure network performance. It is essential to identify anomalous data coming in before they reach the core services. As such, anomaly detection - the task

of identifying events that deviate from the normal behavior - continues to be an important task. However, techniques traditionally employed by industry on real-world data - Density-based spatial clustering (DBSCAN) and median absolute deviation (MAD) - have severe limitations, such as the need to manually tune and calibrate the algorithms frequently and limited capacity to capture past history in the model. Lately, there has been much progression in applying machine learning techniques, specifically autoencoders to the problem of anomaly detection. However, thus far, few of these techniques have been tested for use in scenarios involving multivariate timeseries data that would be faced by telecommunication companies. In Chapter 5, we propose a novel autoencoder based deep learning framework called ERICA including a new pipeline to address these shortcomings and provide a defense that is effective but also does not put a strain on resources. Our approach has been demonstrated to achieve better performance in terms of detection accuracy as well as memory and time cost.

1.3 Thesis Organization

This dissertation is structured as follows. Chapter 2 provides sufficient background on different types of attacks and defenses under consideration. Related works are discussed in Chapter 2 as well. In Chapter 3, we explore the effectiveness of single-image adversarial examples on multimodal sensor fusion models. In Chapter 6, we develop a new fine-tuning technique to provide a quick protection for models against adversarial examples. In Chapter 4, we investigate a previouly unexplored area in poisoning attacks - the frequency spectrum and show how it can affect both attack and defense methodologies. In Chapter 5, we illustrate how to develop a machine learning system in industry to detect anomalies and provide a defense. Lastly. we discuss future works and conclude the dissertation in Chapter 7.

CHAPTER 2

Background and Related Work

This chapter provides the necessary background information and related work in order to properly understand and frame the dissertation. Section 2.1 provides a high-level overview of machine learning systems for those unfamiliar and also provides some key assumptions in terms of terminology that is used in this work. Section 2.1 then goes into some background information on the attacks studied in this work. Section 2.3 then goes in depth into the relevant related works.

2.1 High Overview of Machine Learning Systems

Though it is hard to pinpoint an exact definition of machine learning, one can say that it is a branch of artificial intelligence (AI) dedicated to understanding computer methodologies that leverage data to improve performance on some set of tasks [114]. These tasks span a wide gamut and includes things like detecting spam in emails, classifying images, detecting objects in images, or translating from one language to another.

This dissertation (and most related work in this field) focus on *supervised machine learning* due to the simple fact that supervised machine learning algorithms greatly make up the majority of use cases in the real world. Therefore, for the purpose of this dissertation, when we talk about machine learning, we are referring to supervised machine learning

techniques unless otherwise noted. In contrast to pure unsupervised techniques, a supervised algorithm is one that operates on data with labeled examples. In the spam detection example, a supervised algorithm will run on cases in which there is a corpus of data in which emails are labeled with spam or not spam. A supervised learning algorithm would analyze the corpus of data - called the *training data* - and produce a function that can be used to analyze new, unseen data. This is in contrast to unsupervised learning algorithms that operate on data without labeled examples.

The portion of the learning algorithm in which the model analyzes the corpus of data is called the training phase. The stage in which the function is learned and the model is deployed to analyze on new data is called the evaluation phase. In this dissertation, we talk about works related to both phases of the supervised learning pipeline.

Another key assumption made in this dissertation is that the term *machine learning systems* is used to refer mainly to deep learning systems and neural networks. This is a common trend as deep learning techniques are able to accurately represent more difficult tasks than other machine learning techniques. As an example, in 2012, Krizhevsky *et al*. [82] submitted the only neural network architecture for the 2012 ImageNet Large Scale Visual Recognition Challenge [141] and completely obliterated the competition. The next year, as a testament to its power, every top performer utilized neural networks in their design.

For this reason, deep learning continues to grow as a popular methodology, has seen astonishing accomplishments [30, 57, 91], and in some ways even matches the level of humans [163]. Consequently, the security of deep learning systems is a natural area to focus on when studying machine learning systems, not for this work but other related works as well. However, while all the works posed here operate on deep learning models, it is still important to note that any security concerns and findings found in this dissertation are also applicable to other machine learning techniques and models.

2.1.1 Formal Representation of Neural Networks

Neural networks are a function $F(x) = y$ that operates on input $x \in R^n$ and produces an output $y \in R^m$. The exact nature of the input and output depends on the task. In the case of image classification models studied in Chapters 4 and 6, for example, the input is raw image pixels and the output is a classification of one of m classes. Formally, the function F also implicitly depends on some parameters of the model θ, but for convenience we do not show the dependence on θ unless otherwise noted since for most of our work, the studied model is fixed.

2.1.2 Relevant Machine Learning Tasks

Throughout the background, we discussed various examples in which machine learning systems and neural networks can be used. In this subsection, we go into a bit more depth for the tasks that are studied in this dissertation.

Image Classification Image classification models are utilized in Chapters 4 and 6. The task of image classification is to assign a label or class to an image. It is expected that every class will correspond to exactly one label (not zero or more than one) and the entirety of the image corresponds to that one label. Image classification models take an image as input and return a prediction about which class the image belongs to. Common model architectures used for this task include the VGG model [155], ResNet [57] and Inception [162].

Object Detection The task of object detection involves locating instances of predefined objects in images or videos. Common instances of this include facial detection or image annotation. We look at the effects of adversarial examples on a specific instance of object detection: 3D object detection in Chapter 3.

Anomaly Detection The task of anomaly detection is the task of of identifying unexpected items or events in data sets, which differ from the norm, or the majority of the data and do not conform to a "well defined notion" of normal behavior. The tasks of anomaly detection can be applied in flow studies, finance and banking, and medicine. They are also useful as a front-line defense in detecting adverse and malicious incoming data. As such, it is an important task in many companies. In Chapter 5, we explore the task of anomaly detection as a defense in Ericsson, a multinational networking and telecommunications company.

2.1.3 Representation

In this section, we give a quick background on the types of inputs we are manipulating and concerned with.

Images A majority of the work done in this dissertation is on models that utilize images as inputs. A colored RGB image (such as in CIFAR-10) is represented as a three-dimensional vector $x \in R^{3hw}$ where h and w are the height and width of hte image. We do not convert RGB images to HSV, HSL, or other cylindrical coordinate representations of color images as all the models here act on raw pixel values.

3D Depth Data 3D object detection models often run on 3D depth data, for example, collected from a Velodyne LIDAR sensor. The nature of the data varies from dataset to dataset (e.g. polar coordinates versus Cartesian coordinates). For the KITTI dataset [47], which we utilize in Chapter 3, each value in the dataset consists of a vector $x \in R^4$ where the first 3 values are the (x, y, z) coordinates in 3D space and r is the reflectance value.

Time Series Data Time series data is a sequence of data points indexed in time order, and typically consist of successive measurements made from the same source over a fixed time interval. For example, the price of a certain stock every minute would be considered a time series data. Each point would be a number corresponding to the value. In the case of the work showcased in Chapter 5, the data used is *multivariate* time series data. That means that for every time stamp, the value is in $x \in R^n$ corresponding to the value from n different sources that are usually related.

2.2 Security of Machine Learning Systems

The field of security of machine learning systems took off in the mid 2010s, not coincidentally with the rise of accuracy in deep learning models. Before this case, it was common to see machine learning models have less than desired performance on various tasks, and so the concern of security was not seriously considered. We can see this, for example, with the fact that there were 0 papers on arXiv with the term ”adversarial” and ”machine learning” before 2014.

2.2.1 Adversarial Examples and Attacks on Models during Inference Time

However, as neural networks began to become more popular and achieve accuracies on various tasks that were previously not possible, the area of security started to take off. The field largely started with an almost accidental discovery by Szegedy *et al.* in their seminal paper [164] in which the authors noted that they were able to misclassify an image on an supervised image classification model by ”applying a certain imperceptible perturbation”. They were the first to coin the resulting image as an ”adversarial image” or ”adversarial

example" and the name stuck. In short, an adversarial example is any purposefully crafted input that is designed to cause an erroneous output in a machine learning model. Though it was originally discovered in images, adversarial examples can exist in a variety of tasks including speech and text.

More formally, the definition of adversarial examples is as follows. Suppose we have a learned classifier (mapping) $F : X \to Y$ from an input domain X (like an image) to a set of outputs Y (like a classification label). Given a benign example $x \in X$ and corresponding proper output $y \in Y$, an adversarial example is a deliberately modified input $x_{adv} \in X$ such that the classifier outputs $F(x_{adv}) \neq y$ (called an untargeted attack) or $F(x_{adv}) = y'$ (called a targeted attack) where y' is a class chosen by the adversary.

A key component of adversarial examples is that the perturbation or change to the input be "minor" - it would be unsurprising to see that any arbitrary large change could cause an error in the output. The term "minor" is largely subjective and differs from task to task - for example, what is considered "minor" in object detection might be different from what is considered "minor" in speech recognition. Even within the same task, the term may change based on the application and domain in question, but for image classification, a commonly accepted term is visually imperceptible to the human eye. So, in the above example, a human should be able to identify x_{adv} as the original output y. Figure 2.1 shows a famous case of an adversarial example for an image.

Shortly after, in another seminal paper, Goodfellow *et al.* [48] explored the nature of adversarial examples, noting that they are likely caused by the linear nature of deep learning models. They also propose a quick way to generate adversarial examples, which became known as the Fast Gradient Sign Method (FGSM).

Since then, works have launched in roughly four directions: creating stronger attacks, creating stronger defenses, understanding the nature behind adversarial examples, and using adversarial examples in various applications. The notable attacks during this time pe-

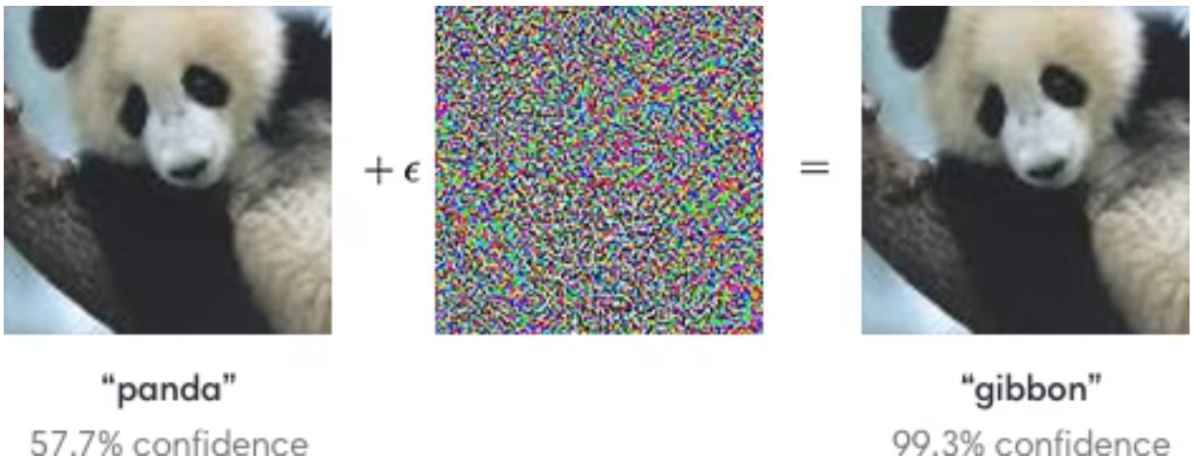


Figure 2.1: An example of an adversarial image from the works Goodfellow *et al*. [48]. Note that the adversarial image on the right causes the classifier to output the wrong class (gibbon) and the resulting image's perturbation is basically impossible to notice to the human eye.

riod were a variation of FGSM [85], Jacobian-based Saliency Map Attack (JSMA)[130], DeepFool [118], the Carlini-Wagner attack [16] and the projected gradient descent (PGD) attack [109]. The latter two are especially noteworthy because not only were they state of the art attacks that broke existing defenses, but they provided a methodology utilizing gradient descent to reliably create adversarial examples. Since then, the works involving adversarial examples have exploded with domains such as malware classification [51], speech [198, 17] We list a concise version of the relevant works related to this dissertation in the Related Work section (Section 2.3).

Defenses In security, there is rarely such a thing as a perfect defense - given enough time and resources, an adversary would be able to thwart almost any defense. The key is to ensure that the defense causes the attacker to expend so much time and resources that either the attack becomes impractical, too difficult, and/or ends up becoming out of scope for the treat model. For example, we consider modern-day encryption to be secure because

assuming a proper key generation, an adversary would need an impractical amount of time to break it.

Security of machine learning systems against adversarial examples follows the same idea. However, even to this day, defenses against adversarial examples is lagging. Early signs of progress were quickly shown to be ineffective as soon as iteration based gradient attacks like CW and PGD were discovered and adapted to try to overcome defenses. These early defenses include obfuscating gradients [4] and regularizing logits [120]. Furthermore, it has been shown that combining weak defenses doesn't automatically make them strong [58].

Most defenses today fall into two major categories. The first - and most ideal solution - is to utilize defenses called certified robustness, which provide a *provable* guarantee that the model will not misclassify an input given that the distortion is within some bounds. These can be either deterministic [181, 113, 49, 32] or utilizing randomized smoothing [31, 89, 92]. However, despite the immense amount of progress made, the bounds that are guaranteed today are still relatively small, and not yet useful in a practical, real-world setting. For example, Lecuyer *et al*. [89] can only certify a 61% accuracy on the CIFAR-10 dataset with an l_2 perturbation radius of 0.25

Because of this, many models rely on the second type of defense called adversarial training which provide an *empirical* defense. The goal of adversarial training is to train a model from scratch to not only perform well on the unaltered, benign data but also perform well under adversarial attacks. The ending result is a model that *usually* performs slightly worse on unaltered, benign inputs (benign accuracy) as compare to its vanilla model counterpart, but makes up for it by performing substantially better on adversarial inputs. Note that, the debate whether robustness and clean accuracy are inherently are at odds with each other is an ongoing debate. Some argue that the two are an unavoidable tradeoff [171, 138] while some argue the opposite [158, 137, 191].

At a high level, adversarial training alters the objective function such that instead of simply trying to minimize the loss, the objective function is rewritten as a minimax problem in which in each epoch of training, the "adversary" tries to maximize the training error with a malicious input and then the "user" tries to minimize the training error on that input. We consider adversarial training as a potential defense against adversarial examples in sensor fusion models in Chapter 3. We examine adversarial training in detail, describe its shortcomings, and post an alternative approach in Chapter 6.

2.2.2 Attacks during Training Phase

As opposed to adversarial examples which involve attacks during inference time, poisoning or backdoor attacks aim to attack models during the training phase. The high level idea is that an adversary would inject malicious inputs into the training set (or alter existing inputs) in order to cause harm to the model. In the case of poisoning attacks, the malicious inputs cause the model to perform weakly, usually by causing the decision boundary to shift.

Our project focuses on backdoor attacks first proposed by Chen *et al.* [28]. Backdoor attacks are more nefarious attacks in which the malicious inputs are crafted with a trigger (either visible or not) and usually a desired label. After a successful attack, the resulting model maintains good accuracy on normal data and operates as intended. However, when an input is given with the exact same trigger, the model will output the erroneous result.

More formally, suppose we have the a machine learning model trying to learn the function $F : X \rightarrow Y$. An adversary would choose a target $t \in Y$, a trigger or key k and some function K that maps the key onto the domain X to form x^k. This mapping could be as simple as putting a yellow square on a corner of the image or as complicated as using GANs. The goal is to make the confidence of $F(x^k) = t$ high while making sure that any pair not associated with the key maintains the same accuracy.

As opposed to poisoning attacks, [77, 124, 132, 7, 1, 9, 8, 98, 111, 184, 190, 157, 99], backdoor attacks are more stealthy and harder to detect (hence the name) since they do not alter the behavior of the models on inputs without the trigger. They also require a smaller percentage of the training data to be modified in order to be successful - standard procedure is less than 10% of the data while poisoning attacks modify anywhere from 20-50%.

Backdoor attacks have been realized in many types of models and applications [76, 180, 129, 127, 128], and just as before, we list relevant works in this field in Section 2.3

2.3 Related Work

We summarize the related works in several categories below

2.3.1 Adversarial Attacks on Sensor Fusion Models

Since the idea of adversarial attacks took off, researchers have been looking at different ways it could be applied, among which were object detection models. Adversarial attacks on object detection models (like YOLO) include attacks in the raw pixel space [185] to launching these attacks in the physical world [40, 202, 64, 167]. We draw inspiration from these techniques when attacking the 3D object detection model. However, some of this work is not directly transferable and so we modify the attacks to fit AVOD.

Because we are interested in modifying a single input in a sensor fusion model, we could have also modified the point cloud input instead of the image input. In fact, there exists prior work attacking 3D object detection models have largely been aimed at attacking models that solely use point cloud data [12, 179, 160, 173]. However, in the physical space it is more difficult to launch an attack to fool the LIDAR sensor. For this reason, we are motivated to look into attacks that solely modify the image, as as this type of attack is much easier for an adversary to perform.

A related, but slightly orthogonal paper explores methodologies to make AVOD more robust against single source distortion [78]. However, they explicitly do not consider adversarial examples and we analyze their proposals as possible defenses in Chapter 3, Section 3.7. Another work explores adversarial attacks on semantic segmentation models that utilize thermal data and RGB image [194]. However, their results are preliminary and they utilize attacks like FGSM that are considered fairly weak in the community.

The most closely related paper is by Wang *et al.* that explores adversarial examples on models utilizing LIDAR and images [177]. However, their work differs from ours in two crucial ways. First, they do not evaluate attacks on a properly created sensor fusion model whose architecture is conducive to 3D multimodal object detection. Instead, they simply combine exiting architectures - a LIDAR featurizer with a YOLO model, for example. The difference between the architectures is made apparent in the AP scores reported: AVOD (which our paper uses) has an AP of 71.88 while the paper's architecture has an AP score of 60.3. Secondly, whereas the aforementioned work only explores one attack, we are able to develop a wide variety of attacks and delve deeper into the nature of sensor fusion models.

2.3.2 Relevant Backdoor Attacks and Defenses

Backdoor Trigger Generation. The first successful backdoor attacks on modern deep neural networks were demonstrated through the BadNets attack [53], using nature images, and the blending attack [28]. Since then, advanced attacks have been developed to improve the trigger effectiveness and stealthiness [94] as well as with various attacker models, such as inserting the backdoor directly by modifying the model's parameters without accessing the training set [104]. More recently, Sarka et al. [145] proposed to utilize GANs to synthesize triggers to achieve a more robust stealthiness. In our dissertation (Chapter 4), we

analyze all these attacks in the frequency domain and find they all exhibit high-frequency components that distinguish them from their corresponding benign untriggered images.

Backdoor Data Detection Prior work on backdoor data detection has either attempted to identify outliers directly in the input space [46] or analyzed the network response given the input. Peri et al. [133] utilize the deep features of inputs to help detect poisoning labels. [20] discovered that normal and poisoned data yield different features in the last hidden layer's activations; [169] proposed a new representation to classify benign and malicious samples; [81] computes influence functions to measure the effect of each input on the output. Other approaches include using input-saliency maps such as Grad-CAM to detect if a model only relies on a certain portion of input for its prediction [29]. Contrary to prior work, which focuses on mage space or the model response given an image input with ad-hoc designs, we examine backdoor data in the frequency domain, enabling a simple yet effective method to backdoor data detection.

Poisoned Model Detection. Existing work has also explored an approach of discerning if a given model is backdoored. The most recent technique uses a meta-classifier trained on various benign and backdoored models, and it works well even under attack-agnostic situations [189]. Other popular techniques include [175, 65, 21, 54, 103] and are based on reconstructing the trigger from model parameters and performs detection based on the reconstructed triggers. However, they are ineffective for smooth triggers as their reconstruction algorithm often assumes the true trigger is patched locally to a clean image. Our work contributes to this line by demonstrating that these techniques can be further improved by incorporating models that attacked with smooth triggers.

Attack Invalidation. Another approach of mitigating backdoor attacks is to prevent backdoor attacks from taking effect. One way to achieve this is by training an ensemble of

models and take a majority vote of their predictions [90, 74, 75]. Other techniques include using differential private training algorithm [39], and various input preprocessing [105] and data augmentation [10, 196] methods to invalidate backdoors in the model or triggers in the samples. Still others even try to erase the backdoor weakness from the model entirely [96, 97]

Our work does not intend to replace any of these defense strategies. Instead our work is complementary to this line of work as a frequency analysis and contributes to the field of defense techniques in two ways: 1) We highlight the possibility of the existence of smooth triggers and aim to bring attention of the community to modify defense techniques to take into account this possibility and 2) we posit a new detection strategy that utilize only the frequency artifact features to identify if a given image is potentially a backdoor trigger.

2.3.3 Anomaly Detection Methods

When deciding a new anomaly detection implementation, there are many prior works we could consider drawing ideas from.

Traditional Anomaly Detection Methods Many different techniques have been proposed for the task of anomaly detection. These include the use of principal component analysis [87, 95] and various clustering algorithms. Other works for anomaly detection include isolation forests [101], and in the realm of machine learning: support vector machines (SVMs) [110] and random forests [100]. However all of these works run into the problem that while they are relatively simple to use, they are not robust enough to handle complicated large scale multivariate time series that are found in the real world.

Deep Learning Methods With the advent of powerful deep learning models, works exploring their use in AD also increased. Recent works have suggested applying Gaussian

Mixture Models [208], Generative Adversarial Networks (GANs) [93, 205] and adversarial nets [204] to the task of anomaly detection. However, these techniques pose the opposite problem that they are too complicated for real-world industry situations in which memory and speed are important limitations.

Autoencoders for Anomaly Detection As shown in Chapter 5, we decided to use an autoencoder as the core structure for our detector. The idea of using autoencoders for anomaly detection in real world systems is widely studied. For instance, [126] used autoencoders for supply chain management while [166] used autencoders to detect anomalies in patient electrocardiogram readings. Other areas that in which autoencoders are used are in gas turbines [45] and even motor sports [174]

However, the use of autoencoders in AD for real-life telecommunications data is yet to be studied. A similar work that looks at time series using autoencoders is [112] but unfortunately requires access to labeled training data. In a similar vein, [206] used autoencoders to improve anomaly detection in a weakly supervised settings. Unfortunately, none of these works directly apply to our specific scenario, either due to a difference in data types or the type of learning assumed (unsupervised versus supervised) ERICA's pipeline and model architecture are customized specifically to suit the nature of large-scale, unlabeled time series data and the problems faced.

ERICA's pipeline and model architecture are customized specifically to suit the nature of large-scale, unlabeled time series data and the problems faced by the network and telecommunications industry.

2.3.4 Adversarial Fine Tuning

Adversarial Training and its Variants Since the basic idea of using a minimax training objective to harden a model (adversarial training), was proposed [109], there have been

many suggested ways to improve it. These include modifying the objective function [200, 71], utilizing unlabeled data as part of the training procedure[18], or using adversarial weight perturbations[182]. However, all these methodologies all involve training a model from scratch.

Other works try to improve a model against adversarial examples in other ways. Similar works include [23] which treats adversarial training as a semi-supervised problem and [186] which uses an adversarial momentum contrastive learning method to save time. Other approaches include identifying and hardening weak subnets [55]. However, none of these works address the issue of fine-tuning and our work achieves a faster result because we are not training a model from scratch.

Fine Tuning a Model In a slightly orthogonal approach, there have been fine-tuning approaches to defend a model against a different type of attack: backdoor or poisoning attacks [150, 102, 122]

The most similar work is by Jeddi et al [73], in which the authors propose taking a pre-trained model and running adversarial training for a few epochs. The authors report similar accuracies on adversarial inputs and benign inputs as would be achieved under regular adversarial training. However, the time to train the final model is much less with their approach. To do this, the work utilizes a "slow start, fast decay" strategy to help the model gain robustness while maintaining a good accuracy on the previous learned clean examples. Since this work is the closest to our work and is considered state of the art, further details are given in Section 6.2.

A similar line of work by Croce and Hein [34] are also interested in fine-tuning but their approach involves taking an already robust model and making it robust to other norm attacks.

In our work, we take this idea further and suggest a different training methodology that

slightly increases training time but results in higher accuracies. Our work also has better accuracy than a methodoloy proposed by Hendryks *et al*. [61]. It is important to note that our work still takes much less time than regular adversarial training.

Transfer Learning We can look a bit further and examine papers related to out-of-distribution inputs and transfer learning. One can argue that the problem of adversarial fine-tuning is a very specific case of transfer learning. However, many papers directly related to transfer learning of adversarial examples [19, 62] provide methodologies that are too complex and needlessly adds complexity and an increase in computational resources.

A problem that is related is the question of the nature of adversarial examples, and how they are related to the larger umbrella of out-of-domain inputs. For example, Kirichenko et al [80] suggest that simply retraining the last layer is enough to force the model to rely less on background, spurious correlations. Other works mention pre-training [170] to increase accuracy, but do not mention adversarial examples However, we find that adversarial examples do not behave in this way. One notable work is by Kumar et al [84], who have proposed a methodology that works better than simple fine-tuning. Their goal is to train models on inputs that are completely different from the ones trained on, for example CIFAR-10 to STL or CIFAR-10 to CIFAR-10.1. Another work suggests the ideas in our paper are possibly by examining which layers are most susceptible to adversarial examples [153]. Though they do not provide a strict training regimen, their work for the most part, agrees with our findings. We apply these ideas to adversarial training and run empirical tests on the effectiveness.

.

CHAPTER 3

Adversarial Examples on Sensor Fusion Models

3.1 Introduction

The ubiquitous use of machine learning systems for different use cases brings about a difference in the types of models use. For instance, while a camera based company might use machine learning models based on image inputs, a translator company might utilize models based on text or speech based inputs.

A different type of model that is gradually coming into use are multimodal models that take in, not one, but multiple inputs. For instance, Meta launched an open source, deep learning framework for vision and language multimodal research [135].

The type of model we are interested in for this project are sensor fusion models. These types of models are multimodal 3D object detection models that take in two types of inputs: a 2D image from a camera and 3D depth data usually from a LIDAR sensor. These kinds of models might be utilized, for example, in autonomous vehicles (AVs) to help detect the environment around them. With the growing proliferation of autonomous vehicles, their security is becoming more paramount, especially against adversarial examples [202, 160].

It has long been known in the community that machine learning models are vulnerable to adversarial examples, maliciously crafted inputs designed to intentionally fool the model into outputting an erroneous result. These range from attacks in the raw pixel space [185]

to launching these attacks in the physical world [40, 202, 64, 167].

All the above attacks are on detector models that utilize single input. There is a belief that the use of additional inputs can mitigate the effect of adversarial examples. While recent work [78] has shown theoretically that models that take in multiple inputs are vulnerable to potential perturbations in a single input, no one has actively explored the robustness and crafted adversarial examples against sensor fusion models. Our work is the first to demonstrate the insecurity of sensor fusion models to several realistic adversarial attacks for 3D object detection.

Though there are many multimodal 3D detection architectures available, we focus this study on models that take in the two types of inputs simultaneously. We purposefully choose to ignore model architectures such as Frustum-Pointnet [136] that utilize a "pipeline" structure in which image data is taken in first followed by LIDAR data because these models are trivially vulnerable to image-based attacks — any existing attack algorithm to fool 2D image object detectors will be able to fool the entire model.

Instead, for this work, we choose AVOD [83], an open-source 3D object detection model, because of its near-top performances among open-source models in the KITTI benchmark. AVOD takes in LIDAR point clouds - which are featurized into BEV point clouds, and RGB images as input. It consists of a two-step detector architecture typical of sensor fusion models. The first stage consists of the region proposal network (RPN). The second stage takes in the proposed regions from the RPN and outputs a bounding box and appropriate classification. Both sub-networks utilize post-processing non-maximum suppression (NMS). Its two stage detector network architecture is one that is typical of sensor fusion models, making it an ideal candidate for our study.

Because this is the first foray into this field, we assume that the adversary is a white-box attacker, having full access to the model. Despite this, in order to guide future research works, we aim to be as realistic as possible; this includes restrictions that the adversary

will not be able to modify the model arbitrarily, including any post-processing steps.

Our key contributions are as follows:

- We perform the first study of adversarial examples on proper sensor fusion models for 3D object detection. We modify existing techniques to show that sensor fusion models are vulnerable to adversarial attacks that modify just the image input. These attacks include the *raw pixel disappearance attack* (94.17% success rate) and a spoofing attack (89.1%). We then analyze the model architecture to show that despite the symmetric architecture, the model frequently leans heavily on the LIDAR input to detect obstacles.

- Building upon the raw pixel disappearance attack, we develop a new methodology of constructing generalized adversarial examples in which one single noise can fool many samples.

- We develop a spoofing attack that can trick the model into detecting objects (vehicles, pedestrians, and cyclists) that should not be detected. We develop two versions of this attack: one to spoof a location already proposed by the RPN a second to spoof an arbitrary location . The first version is able to achieve a success rate of 89.1% while the second is only able to achieve 9.7%. To explain the reasoning, we show that despite the symmetric architecture, the model frequently leans heavily on the LIDAR input to detect obstacles.

- We explore some basic defenses, including robust training and a novel fusion layer [78]. We comment on their effectiveness and put forth suggestions for future directions.

3.2 Threat Model

For this work, assume that the adversary is a white-box attacker, having full access to the model. Despite this, we limit the adversary to modification of just the image. This is because we believe an attack through modification of just the image is much easier to carry out in the real world than an attack that requires modifications to the LIDAR sensor, which is supported by several works [12, 160, 173].

Thus, by restricting attacks to just images, we are assuming a less powerful and more realistic attacker. Finally, we add another restriction that the adversary will not be able to modify the model in any way, including any post-processing steps, like NMS.

3.3 Disappearance Attack

In this section, we explore attacks that are able to fool the model into not detecting an object it had previously detected (*raw-pixel disappearance attacks*) and those that fool the model into detecting an object that is not actually present (*spoofing attack*).

We first try an attack methodology based on projected gradient descent [109]. This is a similar attack as was analyzed in Wang *et al*. [177]. We test this technique on 220 different samples and are able to achieve a success rate of 84.7%. However, the main concern was that the attack was visually noticeable. Though there are application scenarios in which this is acceptable, we then challenge ourselves to construct adversarial examples that could accomplish the same goal but achieve better ”stealthiness”.

The *raw-pixel disappearance attack* is motivated by a desire to create a disappearance attack that results in an adversarial example that is less noticeable to the human eye. As stated in the threat model, we limit the adversary to modifying just the image. In order to make our attack more realistic, we have also disallowed the adversary from looking at

any values before any post-processing steps. We explore a different kind of attack - patch attacks - in Section 3.4.

To cause the desired object to disappear, we aim to remove potential bounding boxes around said object. Removing a bounding box can be done by forcing the output softmax probability of an object to fall below the detection threshold. We will call this set of all potential boxes that we need to attack B. For ease of notation, suppose $C(w, b) \in R^c$ denotes the output classification softmax of bounding box b on image w and $C(w)$ outputs all the potential bounding boxes of image w in decreasing order according to softmax score- in other words, the first element of $C(w)$ is the bounding box with the highest confidence.

To find an adversarial noise δ, we will attempt to solve the following function:

$$\text{argmin}_\delta \sum_{b \in B} F(C(w + \delta, b)) \tag{3.1}$$

where $F(.)$ represents use of the softmax. Since we wish to make the perturbation to the image as small as possible, we add another element as suggested by the CW attack [16], $D(w + \delta, w)$, which measures the L_2 norm between the adversarial image and the regular image.

Thus, the final loss function $L(.)$ that we are trying to minimize becomes:

$$L(w + \delta, B) = \sum_{b \in B} [C(w + \delta, b)] + \epsilon * D(w + \delta, w) \tag{3.2}$$

ϵ is used to weigh one value versus the other. The optimal value of ϵ is found through binary search.

Unlike previous work, we choose to attack the second-stage detector (instead of the first stage RPN) as it results in an attack with less distortion.

There lies an additional challenge in the fact that due to NMS and the restrictions we set

on the adversary, not all of the elements of B will be visible. In other words, not only do we not know the set B a priori, for some bounding boxes $b \in B$, $C(w, b)$ is not existent and the logits will not be visible. For some related previous work, this was not a huge limiting factor [66] while others went around this by trying to attack the NMS algorithm itself or to modify the NMS threshold to obtain all bounding boxes [185].

Since neither solution is allowed under our threat model, we instead modify the algorithm to greedily attack the top confidence bounding box that is visible - as we keep trying to lower the confidence of the bounding box with the highest score, one of two outcomes will happen. In one, the object in question will no longer be detected, in which case our attack goal is accomplished. In the other case, the bounding box in question will be removed via NMS and the next top-score bounding box will appear and the process can be repeated. This process will remove all objects present in an image, but an adversary can selectively remove certain objects by applying a mask. In this case, the objective function needs to be modified to attack the top k bounding boxes simultaneously. The full algorithm is shown in Algorithm 1.

Algorithm 1: Raw-pixel attack

input : Raw image w, k
output: Adversarial noise δ
begin
  $\delta \leftarrow 0$
  while *Object is still detected* **do**
    $B' \leftarrow C(w + \delta)[0...k]$;
    $\delta \leftarrow argmin_{\delta} L(w + \delta, B')$
  end
  return δ
end

Figure 3.1: Results of some of our raw-pixel attacks. The left images are outputs for benign images. The 1st value corresponds to the classification confidence and the 2nd value corresponds to the IOU with the ground truth bounding box. The right images show the corresponding adversarial images. Note that our attack works in deleting any number of objects, whether they are in the foreground or background (the attack shown in the bottom right image purposely targets one vehicle and not the other). The red boxes are ground truth and the green boxes are bounding boxes outputted by the model.

3.3.1 Evaluation and Results

When training the AVOD model for this and future evaluations, we follow the methodology followed in the paper proposed by Chen et al [26] and split the trainval set into a training set with 3712 samples and a validation set with 3769 samples for better performance. We train all models to closely match the results stated in the original paper.

We start off with an instance of the AVOD model designed to detect vehicles. We then choose 3000 random samples containing a total of 10,920 detected vehicles and try constructing adversarial examples using the method stated above. We are able to achieve a 94.17% success rate. Some of the attacks are shown in Figure 3.1. Note that the the adversarial perturbations are difficult to notice.

Unsurprisingly, we find that vehicles in the foreground require more distortion than vehicles in the background. To normalize this, we divide the L2 norm by the area of the

vehicle we are targeting. The amount of distortion needed is shown in Figure 3.4.

To further explore the effectiveness of the raw-pixel disappearance attack, we run the attack on another instance of AVOD that identifies pedestrians and cyclists, using 7,585 pedestrians / cyclists. We are able to achieve a 97.11% success rate on the objects.

3.4 Towards Generalizability

In this section, we attempt to create a single adversarial patch that, despite being more noticeable to the human eye, would be able to be universally applied to any vehicle and cause them to escape detection from the model. This is a key step in determining the feasibility of attacks in the physical world.

To start, we draw inspiration from the expectation over transformation (EOT) algorithm [5]. In the case of the KITTI dataset however, it is difficult to apply any transformation to an image and also properly modify the corresponding LIDAR data. Therefore, we decide to use different object samples available in KITTI instead. For each image, we identify an area to apply the patch (e.g. on the vehicle). Note that if an image has multiple objects, we may have to apply the patch separately to different areas. For the sake of simplicity, we consider these as separate images; if an image has two vehicles and we wish to apply the patch separately on each of the vehicles, we represent this instead as two separate images, each with just one vehicle.

Let $P(w, \delta)$ be the operation that applies adversarial patch δ to image w, appropriately resizing the patch as necessary. If we have a set of images T (along with their corresponding bounding box set), we would be able to create a universal patch by solving the following objective function:

$$\text{argmin}_{\delta} \mathop{\mathbb{E}}_{w \in T}[L(P(w, \delta)) \tag{3.3}$$

Normally, this would be done simultaneously via batching. However, the AVOD model and many other sensor fusion models do not support batching and so we alter the algorithm: instead of operating over all the images simultaneously, we perform the objective function on one image at a time, keeping the noise in between images. We iterate over all the images several times trying to ensure that convergence is reached.

The number of times to iterate is a hyperparameter that must be tuned but for our experiments we iterated 25 times. This is a similar approach as suggested by [116], however, we do not project all perturbations onto a p-norm since we find it slows our algorithm due to the nature of our loss function trying to target every potential bounding box. The resizing and the nature of multiple bounding boxes are also reasons why the GD-UAP[119] is less than ideal for this work. The algorithm can be viewed below (Algorithm 2).

Algorithm 2: Modified EOT

input : Set of images T, k, n, ϵ
output: Adversarial noise δ
begin
  $\delta \longleftarrow$ RandomInit;
  for $i = 0$ **to** n **do**
    $w \leftarrow$ NextImage(T, i);
    $B' \leftarrow C(P(w, \delta))[0...k]$;
    $\delta \leftarrow argmin_\delta L(P(w, \delta), B', \epsilon)$ $\epsilon \leftarrow$ UpdateEpsilon(i)
  end
  return δ
end

Note that the loss function is the same except that it now takes in an additional ϵ as an argument. This is because ϵ might need to be modified after every iteration to ensure convergence (performed by UpdateEpsilon). We find that gradually decreasing the value until a certain floor value works well.

This algorithm allows creation of universal patches with models that do not easily support batching. However, one drawback to this approach is the introduction of additional hy-

Figure 3.2: The left images are the result of the custom adversarial patch and the right images contain a random noise patch applied in the same area.

perparameters. For instance, the algorithm requires the defining of a NextImage function, which outputs the next image to be attacked. Since the algorithm is performed sequentially, the order in which the images are fed affect the strength of the result. Through experimentation, we find that a random ordering of images is sufficient. We leave exploration of the optimal ordering to future work.

3.4.1 Results

We run this algorithm on the validation set and are able to achieve a success rate of 64.03%. To establish a baseline comparison, we apply a random noise patch to the same vehicles. These random noise patches, when applied to the vehicle in the same location, achieve a 0% success rate. The results of this case study suggests a worrisome fact that sensor fusion models are still vulnerable to universal physical adversarial examples, similar to what is shown in Huang *et al.* [64].

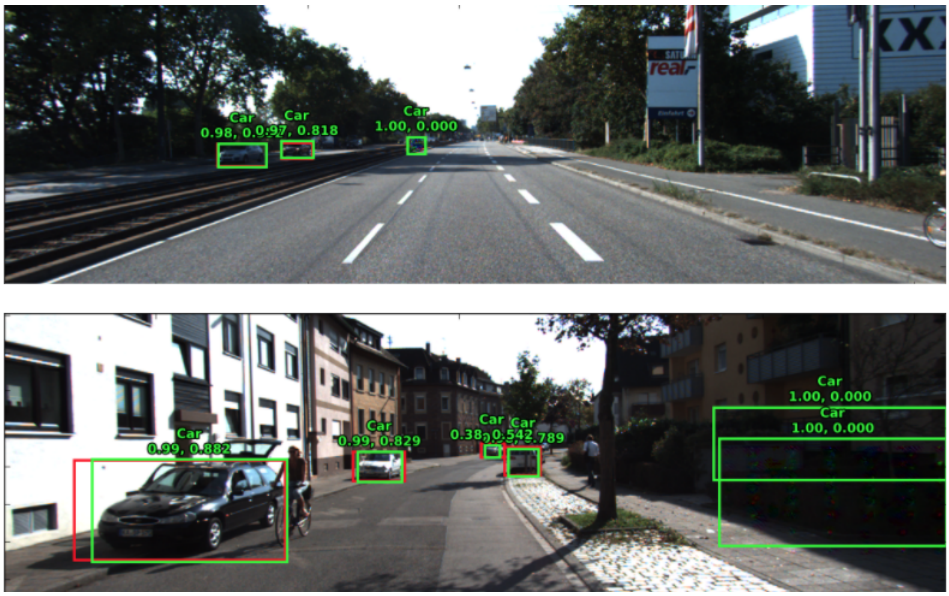


Figure 3.3: Results of some of our spoofing attacks (SpoofV1). The bounding boxes with an IOU of 0 (second value) are the spoofed objects and the red squares represent the ground truth.

3.5 Spoofing Attack

In this section, we discuss a different type of attack we created. Instead of causing a vehicle to disappear, the *raw-pixel spoofing attack* creates adversarial examples that fool the model into detecting an object that is not actually present.

Intuitively, we can take the existing objective function for the raw-pixel disappearance attack and change the sign of the loss function to cause objects to appear. However, the danger to this approach is that it is possible the desired anchor box will be omitted from the RPN output when trying to create the adversarial example. To solve for this, we add another loss to our adversarial objective function that will help ensure the desire bounding box is outputted by the RPN. This is identical to the loss function except that it targets the first-stage RPN. The final loss function is a weighting of the two losses where the RPN loss function is weighted much more heavily:

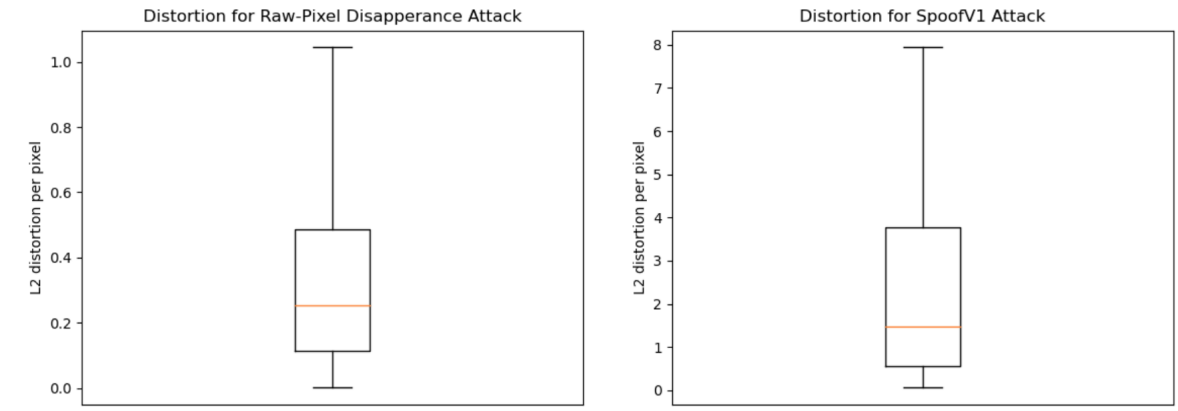


Figure 3.4: The distortion required per pixel in the L_2 norm space for the disappearance attack (left) and the spoofing attack (right) adversarial examples. Note the difference in scale.

$$L_{spoof} = L_{RPN} + \alpha * L_{Stage2} \tag{3.4}$$

We utilize the spoofing attack in two different ways: The first (*SpoofV1*) attempts to spoof a proposed location that is *already* outputted by the RPN. The second (*SpoofV2*) tries to spoof any arbitrary location by fooling the RPN as well so that the desired location is outputted in the list of proposed regions. The only difference between the two attacks is which anchor boxes we feed into L_{spoof}.

As a final point, a very simple defense against spoofing attacks is to remove any anchor boxes that do not have any LIDAR data as a pre-processing step before inputting into the model. Acknowledging this, we run all of our experiments under this assumption to increase the likelihood that our results cannot be defeated by a simple defense.

3.5.1 Evaluation and Results

We run our attacks to create adversarial examples from 440 samples for vehicle detection and 230 samples for pedestrians and cyclists.

3.5.1.1 Spoof Version 1 Attack

For all the samples, we try to spoof 5 random proposal regions under the detection threshold. Our *SpoofV1 attack* is able to achieve a success rate of 89.1% upon evaluating on these samples. Some of the successful attacks are shown in Figure 3.3. We find, however, that compared to the raw-pixel disappearance attack, this attack requires more distortion. To show this, we spoof the 5 region proposals for every image that have the highest score under the detection threshold and take the adversarial example that requires the *minimal* amount of distortion (with respect to L_2- norm) and compare it against our raw-pixel disappearance attacks. We compare these samples and find that the required perturbation for creating a spoofed adversarial example has a wider range. The amount of distortion required for the SpoofV1 attack versus the Raw-Pixel Disappearance attack is shown in Figure 3.4. In general, while there are a few adversarial examples that can be created with a small L_2 norm per pixel, they generally require more distortion than adversarial examples via the disappearance attack.

3.5.1.2 Spoof Version 2 Attack

For the *SpoofV2 attack*, we decide that a spoofing attack on an image is considered successful if it produces an adversarial example that causes an object to be detected with the following two criteria: 1) The detected bounding box has an IOU of 0 with any ground truth bounding box and 2) The detected bounding box is not on any existing object (e.g. a wall or a building). The second requirement is put in place because we imagine an adversary

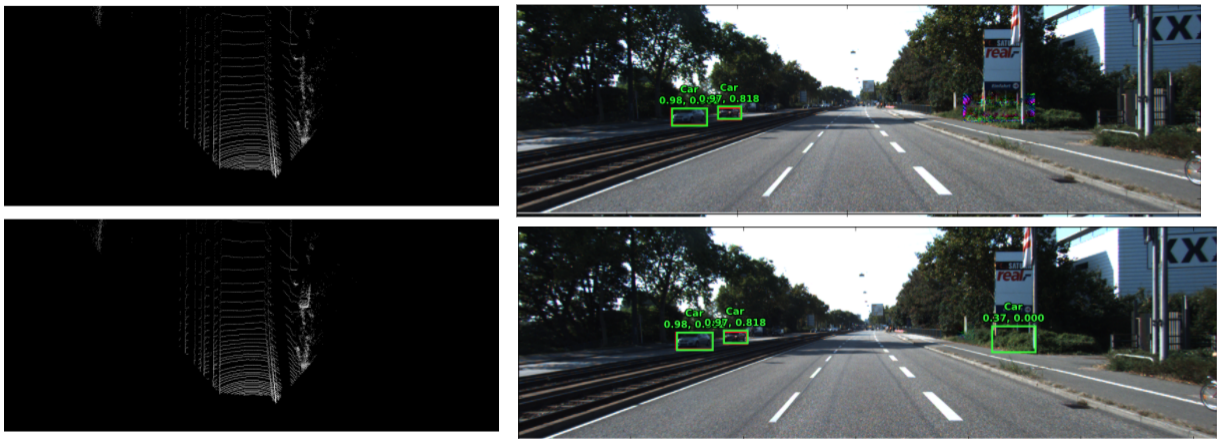


Figure 3.5: The top right image show an attempt to spoof an object underneath the sign. When we modify the LIDAR input (bottom left), we are able to create a successful adversarial example (bottom right).

will accomplish very little by tricking the model that an already-present obstacle is a car (or pedestrian / cyclist). We believe that tricking the model into detecting a vehicle where no object is present may cause more damage. Out of the samples, we are only able to achieve a success rate of 9.7%.

We suspect this is because of the lack of point cloud data in these locations. To test this, we modify the point cloud data and investigate how that affects the ease with which we can create spoofing adversarial examples in the image space. As shown in Figure 3.5, while it is difficult to cause the attack, modification of the point cloud data makes it easier. We modify the point cloud data in the foreground for 300 samples, including the successful samples. With this modification, not only do we find a reduction of the distortion needed by over 25%, but we are also able to create 236 new successful adversarial examples that fit both of the aforementioned criteria. We explore the effects of sensor inputs further in Section 3.6.

Figure 3.6: Output of experiment in which we switched LIDAR and image inputs. The two images on the left show the normal output for benign inputs. The two right images show the output when the LIDAR for one is switched for the other (and vice versa). Note that the model output follows the LIDAR (red box) more than the image.

3.6 Analysis of Sensor Input

Motivated by the results of our experiments on various attacks, we suspect that the model architecture, while symmetrical, heavily utilizes the LIDAR sensor input over the image. To test this, we run an experiment in which we randomly use the LIDAR from one sample and the image for another. This was done for 300 random samples, swapping the image of one and the LIDAR of another, resulting in 90,000 combinations. This experiment helps give an insight of how the model performs when the image and the LIDAR are at odds with each other. Some of the results are shown in Figure 3.6.

For the sake of simplicity, we consider an object as ”correctly identified” if the bounding box was correctly drawn according to the LIDAR sensor. Note that we could have easily swapped and used the image input as ”ground truth” but that would not make any change to the final result. Amongst all the potential bounding boxes, 91% of them were correctly identified, despite having conflicting image data. Furthermore, only 19% of all bounding boxes detected within the 900,000 combinations did not correspond to any ground truth

Type	Disappearance Attack Success Rate	Spoof Attack v1 Success Rate
Baseline	0.94	0.89
Distorted Inputs	0.92	0.85
Training MaxSSN without LEL [78]	0.87	0.81
Training MaxSSN with LEL [78]	0.80	0.39
Adversarial Training	0.63	0.51

Table 3.1: Table showing success rate for various defenses against our proposed attacks.

bounding box.

This experiment strongly suggests that the model favors LIDAR data when detecting objects, which helps explain the difficult in the spoofing attakcks. This also suggests that the use of image in this architecture proves to be an "Achilles' heel": while most of the detection of an object is done using the LIDAR input, a well-crafted image input can override this, thus providing an avenue for adversaries to attack and fool the model.

3.7 Exploring Defenses

In this section we analyze some potential defenses for sensor fusion models that may be able to mitigate adversarial examples. These include training on distorted inputs, a novel fusion layer and training methodology proposed by Kim and Ghosh [78], and adversarial training. For each methodology, we test our raw-pixel disappearance attack and our Version 1 Spoofing attack on the same vehicle samples as in the evaluation for our attacks. All results can be seen in Table 3.1.

3.7.1 Training on More Distortion

We first run experiments to address a common belief amongst some manufacturers that simply training on a wider range of inputs (like fog and snow) will help mitigate adversarial examples. We show that this is not the case. To illustrate this, we train an instance of

the AVOD vehicle detection model on an augmented version of the dataset. To create this modified version, we apply all the distortions to every training image as suggested by Hendryks and Dietterich [60]. We find that this does little to affect the success rate of the two attacks. Training on various input distortions like fog and frost is important, and perhaps essential, for safe performance of AVs. However, our experiment shows that it does not eliminate the threat against adversarial examples.

3.7.2 Robust Training and New Fusion Layer

We next analyze methodologies proposed by Kim and Ghosh [78] that provide robustness against single-source distortion in sensor fusion models. We refer readers to the paper for details, but in short, the authors propose several novel loss functions as well as a new fusion layer called LEL to help protect against noisy distortion. However, it is worth nothing that they have not evaluated on adversarial examples.

We decide to test the two designs proposed by the authors that achieved the best performance under noise: applying the new loss function called MaxSSN with and without the LEL. We train both models according to the specifications shown in the paper and achieve a similarly stated accuracy. We then test our attacks on these models. We find that the models do mitigate against our disappearance attacks, but not to a huge degree. The LEL combined with MaxSSN, however, is quite effective in protecting against our spoofing attack. Unfortunately, it is worth noting that there exists a trade-off as both models suffer in a drop of AP score compared to the original model when run on benign inputs.

3.7.3 Adversarial Training

A popular methodology to make machine learning models more robust against adversarial examples is to utilize adversarial training. While there exist many techniques for adver-

sarial training on image classification models, there is little work exploring the technique for object detection models; the technique is not directly transferable as object detection models have many more stages that need to be made more robust. Nevertheless, we utilize a technique proposed by Zhang and Wang [199].

We find that, unfortunately, the AP score drops from 0.73 to 0.64 after adversarial training. However, the success rate of the adversarial attacks drop as well; the raw-pixel disappearance attack drops to a 63% effectiveness while the spoofing attack drops to a 51% success rate. While these results are not ideal, they do not necessarily eliminate adversarial training as a viable option to provide robustness. On the contrary, these results demonstrate that a better adversarial training algorithm may be able to provide robustness. We leave this exploration to future work.

3.7.4 Other Defense Recommendations

Our results do not necessarily eliminate the possibility of a model-level defense, like adversarial training. However, we believe another avenue to explore is incorporating defenses *outside* the model. In a larger system, it could be feasible to utilize the different sensors, for example, to validate one another before feeding into the final detection model. We leave these possible defenses for future work to explore.

3.8 Conclusion

In this chapter we explore a sensor fusion model's security against adversarial examples. We pick a popular architecture and create new techniques to craft adversarial examples on the image input that can cause objects to disappear or be incorrectly detected. We then propose a new methodology to creating generalizable adversarial examples. We also explore the reasoning behind our results and find that the architecture heavily favors the

LIDAR input more than the image input. Given our findings, we consider the image as an "Achilles' heel" and is a potential avenue of vulnerability against adversarial examples. Finally, we evaluate some proposed defenses and posit some future directions to explore for making senor fusion models more robust.

In short, we highlight an important problem that sensor-fusion models are no less vulnerable to adversarial examples. Though we only evaluate on one model due to the lack of availability of open-source models, our evaluation on alternative fusion layers and training loss functions suggest that other models may also be vulnerable to sensor fusion models. Nevertheless, we urge future work to explore robustness of other sensor fusion models when available for a more comprehensive study to better come up with potential defenses.

CHAPTER 4

Using Frequency for Poisoning Attacks

4.1 Introduction

In contrast to adversarial attacks explored in the previous chapter, backdoor attacks are attacks where adversaries deliberately manipulate a proportion of the training data [53, 28], or the model's parameters [104], to make the model recognize a backdoor trigger as the desired target label(s). In other words, the attacker launches the attack during a model's training time.

When the backdoor trigger is introduced during test-time, the poisoned model exhibits a particular output behavior of the adversary's choosing (e.g., a misclassification).

Backdoor attacks have been realized in various domains including graph clasisfication [140, 183], text classification [36, 25, 161], and malware detection [149]. Prior works have also created successful triggers targeting tasks including Generative Adversarial Models [148, 37, 144], reinforcement learning [79, 178, 192], and federated learning [6]. In the image domain, backdoor triggers have been demonstrated to perform malicious tasks such as converting the label of a stop sign or misidentify a face. Due to the danger of backdoor attacks, great effort has been undertaken to defend against them.

State-of-the-art backdoor triggers are designed to be inconspicuous to human observers. One idea of generating such triggers is to use patterns of commonplace objects [104, 180].

For instance, one could use glasses–commonplace objects appearing in a face image–as a trigger to backdoor a face recognition model, thereby hiding the triggers "in the human psyche." Another approach to generate "hidden" or "invisible" triggers is to inject imperceptible perturbations via solving a norm-constrained optimal attack problem [94, 142] or leveraging GANs [145].

Previous research on backdoor data detection either identifies outliers directly in the image space [131] or analyzes the network activations based on an image input [133, 108, 20, 81]. In contrast, we provide a comprehensive analysis of the frequency spectrum across various existing triggers and multiple datasets. We find that all existing ideas of generating samples contain triggers exhibit severe high-frequency artifacts. We provide a detailed analysis of the causes of the high-frequency artifacts for different triggers and show that these artifacts stem from either the trigger pattern per se or the methodology of inserting the trigger.

Based on these insights, we demonstrate that the frequency domain can efficiently identify potential backdoor data in both the training and test phase. We build a detection pipeline based on a simple supervised learning framework and proper data augmentation as a demonstration. It can identify existing backdoor triggers at a detection rate of 98.5% without prior knowledge of the types of backdoor attacks used. A high detection rate is still maintained even when the data used for training and testing the detector have different input distributions and are from different datasets.

Given that present triggers are easily detectable in the frequency domain, our natural question is whether or not effective backdoor triggers can be designed without high-frequency artifacts (which we will refer to as smooth triggers hereinafter). A straightforward approach to generating smooth triggers is to apply a low-pass filter to existing triggers directly. However, in our experiments, we find this simple approach cannot achieve a satisfying attack success rate while still maintaining stealthiness. To design more effective

smooth triggers, we first formulate the trigger design problem as a bilevel optimization problem and then propose a practical heuristic algorithm to create triggers. Our experiments show that our proposed triggers can achieve a much higher attack success rate than the simple low-pass filtered triggers. We further study the detectability of the triggers and show that existing defense works can benefit from incorporating these smooth triggers into their design consideration. Our experiments also demonstrate that detectors trained over strong, smooth triggers can generalize well to unseen weak smooth triggers.

Overall, our work highlights the importance of the overlooked frequency analysis in the design of both backdoor attacks and defenses. We open-source the experiment codes and welcome the public to contribute to future developments. Our key contributions are summarized as follows:

- We perform a comprehensive frequency-domain analysis of existing backdoors triggers, revealing severe high-frequency artifacts commonly across different datasets and resolutions.
- We present a detailed analysis of the causes of these artifacts.
- We show the effectiveness of employing frequency representations for detecting existing triggers.
- We propose a practical way of generating effective smooth triggers that do not exhibit high-frequency artifacts and provide actionable insights into their detectability.
- We demonstrate that defenses that consider attacks with differenr frequency artifacts can achieve a stronger defense effiency. We show this through a case study with some existing defenses and through augmentation of our own detector.

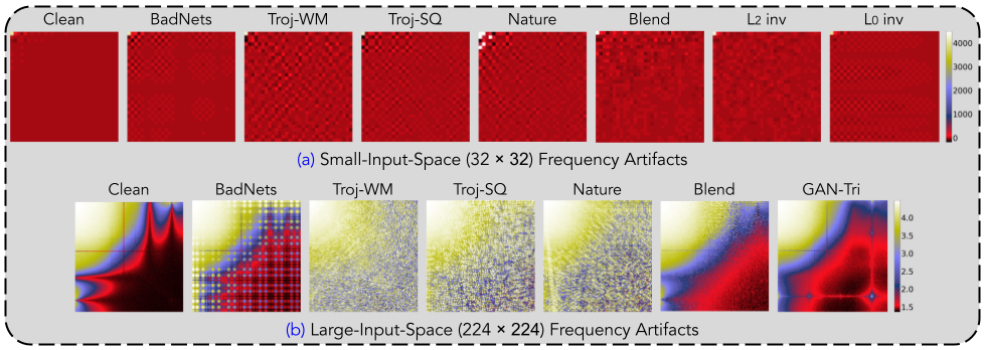


Figure 4.1: A side-by-side comparison in the frequency domain of clean samples vs. samples patched with triggers. The left-most heatmap in (a) depicts the mean spectrum of small-input-space data using 10000 samples randomly selected from the CIFAR-10 dataset. The left-most heatmap in (b) illustrates the mean spectrum of large-input-space data using 1000 samples randomly chosen from the PubFig dataset. The rest images show the mean frequency values of images patched with different backdoor attack triggers. All the frequency results of (b) are depicted from 1.5 to 4.5 using value clipping and exponential calculation for better visualization.

4.2 Frequency Artifacts

Early-stage backdoor triggers were designed to be visible or utilized commonplace objects to hide the triggers [28, 53, 104] or contain visual characteristics that could be easily picked up with visual inspections [28, 53, 104]. More recent attacks can produce triggers that are invisibly designed to human observers [94]. or attack without modifying the original labels [142].

In short, today's backdoor attacks constantly develop the triggers to look as inconspicuous as possible in the **image domain**. We take inspiration from the success of frequency-based GAN-generated fake image detection [44] and examine these existing triggers in the **frequency domain**.

4.2.1 Preliminaries

We utilize the *Discrete Cosine Transform* (DCT) to convert images to the frequency domain. Closely related to the Discrete Fourier Transform, DCT represents an image as a sum of cosine functions of varying magnitudes and frequencies. Our work uses the type-II 2D-DCT, a standard tool adopted in image compression algorithms such as JPEG.

The type-II 2D-DCT is given by a function $D : \mathbb{R}^{N1 \times N2} \rightarrow \mathbb{R}^{N1 \times N2}$ that maps an image data $\{g_{x,y}\}$ to its frequency representation $D = \left\{D_{k_x,k_y}\right\}$ with $D_{k_x,k_y} =$

$$w(k_x)w(k_y)\sum_{x=0}^{N_1-1}\sum_{y=0}^{N_2-1} g_{x,y} cos\left[\frac{\pi}{N_1}(x+\frac{1}{2})k_x\right] cos\left[\frac{\pi}{N_2}(y+\frac{1}{2})k_y\right]$$

, for $\forall k_x = 0, 1, ..., N_1 - 1$ and $\forall k_y = 0, 1, ..., N_2 - 1$, where $w(0) = \sqrt{\frac{1}{4N}}$ and $w(k) = \sqrt{\frac{1}{2N}}$ for $k > 0$.

Similar to previous work [44], we plot the DCT spectrum as a heatmap, where the magnitude of each pixel indicates the coefficient of the corresponding spatial frequency.

The heatmap's horizontal and vertical directions correspond to frequencies in the x and y directions, respectively. The heatmap's top-left region corresponding to low frequencies, and the right bottom area corresponds to higher frequencies. Due to the energy compaction ability of the DCT, the coefficients drop quickly in magnitude when frequencies increase. It is a well-known fact that natural images typically have most of the energy concentrated in the low-frequency section [11, 168].

4.2.2 Examining Images with Triggers using DCT

In this work, we examine the DCT spectrum of the following triggers (which represent the most common and state of the art triggers at the time the work was performed): *BadNets white square trigger* (BadNets) [53], *Trojan watermark* (Troj-WM) [104], *Trojan square* (Troj-SQ) [104], *hello kitty blending trigger* (Blend) [28], *nature image contains semantic*

information as the trigger (Nature) [28], l_2 *norm constraint invisible trigger* (l_2 inv) [94], l_0 *norm constraint hidden trigger* (l_0 inv) [94], and *GAN generated fake facial character as the trigger* (GAN-Tri) [145]. This set of triggers encompasses the two general ideas of designing triggers in existing works: patching visible patters of commonplace objects and injecting invisible perturbations.

Figure 4.1 compares the frequency spectrum between clean images and the images patched with different triggers. The two heatmaps are generated with data sampled from CIFAR-10 (small-input-space) and PubFig (large-input-space). We follow the same settings of [94] and omit l_2 inv and l_0 inv triggers for PubFig. This is because generating l_2 and l_0 invisible triggers requires a pre-trained DNN model to solve the optimization problem, and removed these attacks over large-input-spaces for simplification. Conversely, we omit GAN-Tri for CIFAR because its small input space does not allow effective trigger generation based on GANs.

The left-most heatmaps from Figure 4.1 represent the DCT spectrum over clean data. Multiple classic studies [11, 168] have observed that the average spectra of natural images tend to follow a $\frac{1}{f^\alpha}$ curve, where f is the frequency along a given axis and $\alpha \approx 2$. Similar to previous findings, our results show that the low frequencies contribute the most to the image, and the contribution gradually decreases towards higher frequencies.

In short, in an isochromatic area in the image, one can approximate these areas with a sum of low-frequency functions. However, if a sudden change were to occur, like at the edge of an object, for example, a higher frequency function would need to be used to approxiate the area. Knowing this fact, it makes sense why low-frequency components dominate the frequency spectrum of clean data, since colors mainly change gradually in images and it is rare to see sudden changes in pixel values (e.g., edges in images).

However, in comparison to the spectrum of clean images, images patched with different triggers all contain strong high-frequency components. This fact remains prominent,

even when we evaluate spectral heatmaps for other datasets, including *German Traffic Sign Recognition Dataset* (GTSRB) [156], *Chinese Traffic Sign Database*[1] (TSRD) and the high-frequency artifacts of inserting triggers persist across these datasets as well.

We provide the pair-to-pair comparisons of samples patched with different triggers' visual effects in the image and frequency domain. Figure 4.2, 4.3 illustrate the comparison of the attack cases over the GTSRB and the TSRD dataset. We also provide the pair-to-pair extended comparison of both the image and frequency domain visual effects on the evaluated CIFAR-10 and PubFig dataset in Figure 4.4, 4.5.

[1] http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html

Figure 4.2: A pair-to-pair comparison of clean data and samples patching with different triggers on the GTSRB dataset. The frequency results are averaged over 10000 randomly selected samples from the test set.

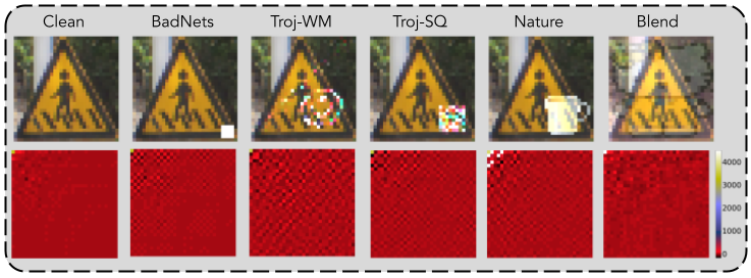


Figure 4.3: A pair-to-pair comparison of clean data and samples patching with different triggers on the TSRD database. The frequency results are averaged over all 4170 samples.

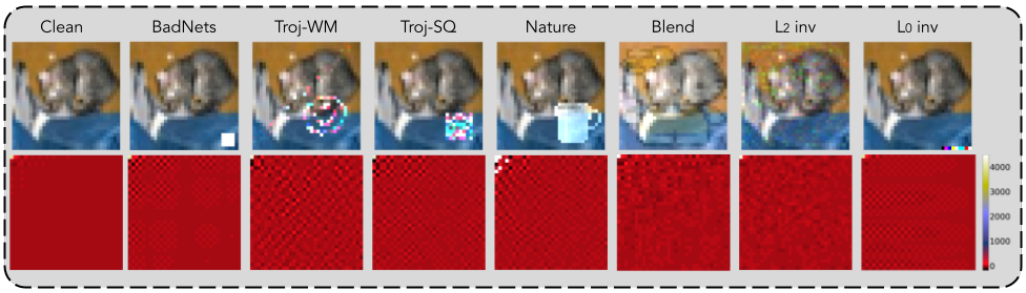


Figure 4.4: A pair-to-pair comparison of clean data and samples patching with different triggers on the Cifar10 dataset. The frequency results are averaged over 10000 randomly selected samples from the test set.

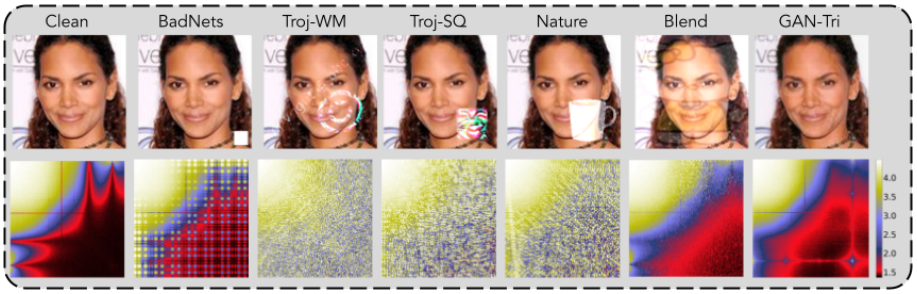


Figure 4.5: A pair-to-pair comparison of clean data and samples poisoned with different backdoor attacks on the PubFig dataset. The frequency results are averaged over 1000 randomly selected samples from the test set and clipped with the range of (1.5,4.5) for visualization.

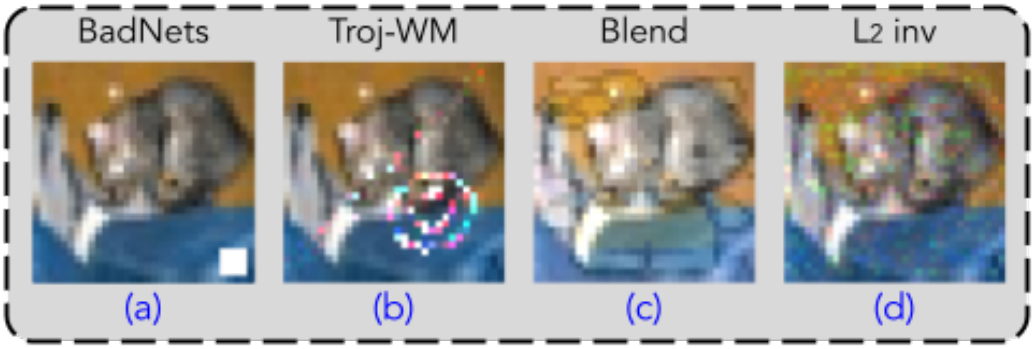


Figure 4.6: Examples of different categories of triggers.

4.2.3 Analyzing Causes of High-Frequency Artifacts

In this section, we investigate the origins of the aforementioned severe, persistent high-frequency artifacts. We examine the causes from two angles, representing two ways of generating backdoor data: additive patching and GAN-based generation. Existing triggers based patching can be further divided into two classes: local patching (e.g., BadNets, Nature, l_0 inv, Troj-SQ) and large-size or global patching (e.g., l_2 inv, Blend, Troj-WM).

Local Patching. Localized triggers can be formalized as $p = T + mask \times orig$, where p is the patched data, T the trigger, $orig$ the original image, and $mask$ is a mask that suppresses the pixel values in the trigger area of the original image. Due to the time-

frequency duality, localized triggers can carry significant high-frequency components per se. By the linearity of DCT, adding a trigger to an image is equivalent to adding the trigger's frequency spectrum to the image's spectrum. Thus, the patched image exhibits a large number of high-frequency components (Figure 4.6 (a)).

Large-Size or Global Patching. For images patched with large-size triggers, their high-frequency artifacts result from either decreased correlation between neighboring pixels or the intrinsic high-frequency artifacts carried by the trigger. For instance, Troj-WM (Figure 4.6 (b)) directly stamps the trigger onto the original data, or $p = T + orig$. Since the trigger pattern has low correlations with the original image's pixels in the trigger's vicinity, one can use high-frequency functions to approximate the patched data. The Blend attack (Figure 4.6 (c)) patches with some small weight use an arbitrary clean image as the trigger. The Blend attack's high-frequency artifacts result from combining two unrelated images, which could induce larger variations of neighboring pixels. l_2 inv (Figure 4.6 (d)) triggers intrinsically are high-frequency perturbations. Thus, patching them onto clean images would directly leave marks in the high-frequency domain.

This fact remains even with clipping. For those values exceeding the normal range, i.e., out of the range of $(0, 1)$ for float data, and $(0, 255)$ for uint8 data, the resulting p would normally go through a value cropping procedure to keep the values of the pixels within the rational range. However, based on the definition of the DCT, this clipping does not reduce the number of high-frequency components. The clipping causes the high-frequency component's value to drop down a small scale that still keeps the significant difference between clean samples and poisoned samples.

GAN-Generated Backdoor Data GAN-Tri utilizes fake facial characteristics generated with GANs (e.g., smiles) to poison the training data and conduct the backdoor attack. Since

a GAN generator maps a low-dimensional latent space to a higher-dimensional data space, upsampling is widely used in GAN architectures. Prior work [44] has shown that the upsampling operations employed in GANs cause inevitable high-frequency artifacts.

4.3 Frequency-Based Backdoor Data Detection

This section describes our experiments to demonstrate the fact that analyzing the frequency domain can effectively distinguish backdoored data from a poisoned dataset. We use the *Accuracy* (ACC) and the *Backdoored data Detection Rate* (BDR) as the evaluation metrics to demonstrate the separability between clean data and backdoor data. A higher BDR means more effective rejection of backdoor samples.

Attacker Model We consider the most potent attacker model, where the attackers have full knowledge of the training set, the inference set, and the potential target model. The attacker can achieve the backdoor attack by either poisoning the training set with samples containing the trigger or directly modifying the target model's weights to insert the backdoor into the DNN. The triggers would then be patched onto the clean samples during the inference time to cause the model to output the target label to complete the attack. This attack scenario is regarded as the strongest attack case.

4.3.1 Detection Method and Application Scenarios

In light of the severe, persistent high-frequency artifacts of existing backdoor triggers observed earlier, we adopt a supervised learning approach to differentiate between clean and backdoor data. To simulate the poison data, we manipulate the clean samples to approximate the high-frequency artifacts that triggers might exhibit. We then create a training set that contains DCT transformations of clean samples and samples with digital manipula-

tions. The digital manipulations used to alter the clean samples include: **1)** random white block: patching a white rectangle of random size onto a random location of the image; **2)** random colored block: adding a rectangle of random size and random value to a random place; **3)** adding random Gaussian noise; **4)** random shadow: drawing random shadows of random shape across the images; **5)** random blend: randomly selecting another sample from the dataset, multiplying it with a small value, and patching with the current data. These perturbations are chosen because they follow the same general methodology as the backdoor attacks.

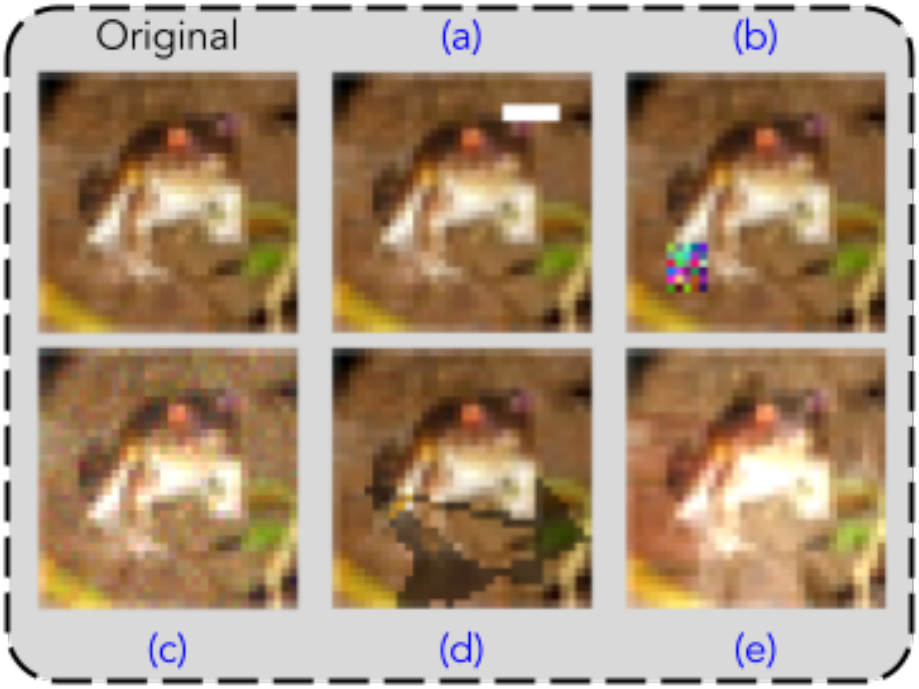


Figure 4.7: Visual examples of the random perturbations adopted in developing the detector. The upper left sample is a clean example, (a)-(e) are the perturbed results using different approaches.

Figure 4.7 presents the visual examples of the random perturbation results. Figure 4.7 (a) is the example of patching a white rectangle of random size onto a random location of the image; Figure 4.7 (b) is the result of patching a rectangle of random size and random

value to a random place. Those two random perturbations simulate patching localized triggers as mentioned and analyzed in Section 4.2.3. Figure 4.7 (c) is the visual result of adding random Gaussian noise; the result of drawing a random shadow of random shape is depicted in Figure 4.7 (d); finally, 4.7 (e) shows the visual result of random blend.

Note that the random perturbations as illustrated here are of different shape and values from the tested triggers. We only use those random perturbations to simulate the resulting high-frequency artifacts using the two major patching methods,

The detector based on frequency artifacts can be applied to both attack scenarios: poisoning the training set or directly tuning the weights. We focus on developing an accurate trigger data detector that can effectively reject triggers during inference. For the scenario where triggers are used to poison the model during training, the detector can also be deployed during training to reject potential poisoned data. We aim to build an attack agnostic detector with zero prior knowledge of the trigger pattern or the target model in both scenarios. This defense case is the most comprehensive scenario aiming to thwart existing backdoor attacks in a trigger-agnostic manner.

When building our detector, we consider the difference in input space and study small input spaces (e.g., CIFAR-10) and larger input spaces (e.g., PubFig) separately.

We test the F1-score and the linear models' overall accuracy on detecting triggered samples using different-input-spaced PubFig datasets. We test ten different values ranging from 32 to 224. The relationship between the input width and the detection efficiency is depicted in Figure 4.8. We can tell from the results that a larger-input-space samples can more easily be used to conduct a linear separation of the benign samples and the triggered samples. We find that attack triggers in larger input spaces (larger than 160 pixels

Meanwhile, the small-input-spaced samples are harder to be separated with linear models. Intuitively, we conduct the DCT of the whole image, thus acquiring a result of the same size as the image domain. So the larger input-spaced samples have more pixels rep-

resenting the high-frequency coefficients, thus better reflecting the high-frequency artifacts when triggers are introduced. Based on the results shown in Figure 4.8 and as claimed in Section 4.3.1, an input space larger than 160 pixels can help linear models meet satisfying detection results.

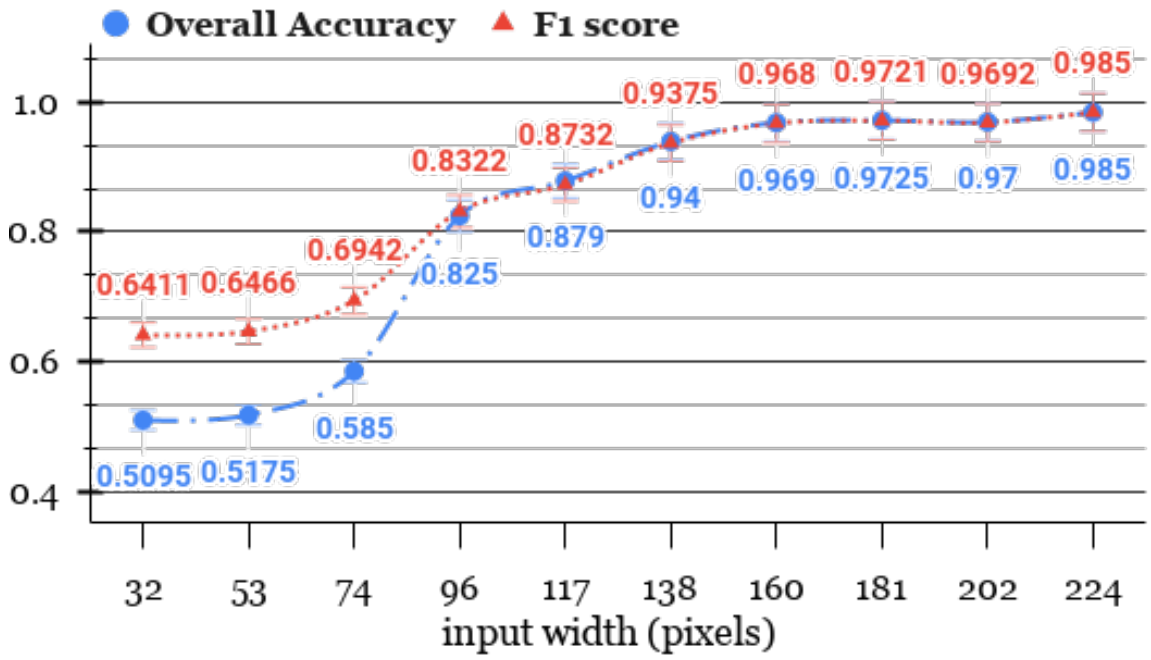


Figure 4.8: Detection Efficiency Using the Linear Model vs. Input Width

4.3.2 Results & Comparison

Experiment Setup. This section evaluates the detection framework assuming we have full access to a clean dataset with a similar distribution as the inference data. In the following subsection, we compare our detection framework's results across different datasets. We use the full original training set for each experiment to develop the DCT processed dataset consisting of equal clean samples and randomly perturbed samples. The test set is consists of half clean samples and half poisoned with the backdoor attack trigger to evaluate the detector's efficiency (e.g., BadNets, Nature). None of the triggers evaluated in the test

set are present in the training set. Table 4.1 shows the results for the CIFAR-10, GTSRB, and PubFig datasets There are 100,000 samples (half clean, half randomly perturbed) in the regenerated CIFAR-10 training set and 20,000 samples in the test set; the regenerated GTSRB includes 70,576 training samples and 25,260 test samples; 22140 training samples and 2,768 test samples for the regenerated PubFig. Results when distinguishing samples in the image domain without DCT were also included as a comparison group. Further details and models used can be found in the Appendix.

	BadNets	Troj-WM	Troj-SQ	Nature	Blend	l_2 inv	l_0 inv
ACC	**94.10**	**98.85**	**98.76**	**98.66**	**97.00**	**98.85**	**98.86**
BDR	**90.50**	**99.99**	**99.82**	**99.61**	**96.30**	**99.99**	**100**
ACC*	49.76	85.17	55.37	54.19	64.52	77.31	49.08
BDR*	1.38	72.19	12.59	10.24	30.90	56.46	0.00

	BadNets	Troj-WM	Troj-SQ	Nature	Blend	l_2 inv	l_0 inv
ACC	**90.23**	**93.96**	**93.93**	**91.46**	**93.67**	**93.96**	**93.93**
BDR	**92.55**	**100**	**99.94**	**95.00**	**99.43**	**100**	**99.94**
ACC*	48.92	57.43	48.61	49.35	80.63	89.53	48.40
BDR*	17.42	31.51	16.92	18.15	69.91	84.65	16.57

	BadNets	Troj-WM	Troj-SQ	Nature	Blend	GAN-tri
ACC	**97.74**	**99.29**	**99.29**	**99.29**	**99.29**	**93.96**
BDR	**96.94**	**100**	**100**	**100**	**100**	**100**
ACC*	53.05	52.55	57.35	60.29	62.27	50.27
BDR*	72.27	72.40	82.01	87.90	91.80	68.30

Table 4.1: The detection efficiency and comparisons on CIFAR-10 (top), GTSRB (middle) and PubFig (bottom) (%). *represents the comparison group using the image domain data.

Results. A supervised detector built in the frequency domain leads to a high BDR (98.5 percent averaging), as shown in Table 4.1. However, the image domain detector (represented by * in Table 4.1) does not work well. We observe an increase in the BDR but a drop in the average ACC using the image data from the PubFig dataset versus the other two, indicating that the BDR improvement on the PubFig dataset causes a higher false-positive rate.

Remark 1. *HWe find severe high-frequency artifacts across existing backdoor triggers which can be utilized to provide accurate detections. Compared with the image domain, the*

Model	#Parameters	Train ACC	BadNets		Troj-WM		Troj-SQ		Nature		l_2 inv		l_0 inv	
			ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR
Linear	6,146	83.35	53.85	28.41	89.64	**100**	89.42	**99.56**	89.57	**99.85**	89.64	**100**	64.65	50.00
128-cell-hidden	393,602	88.23	54.80	21.89	**93.85**	**99.99**	**93.44**	**99.16**	**93.71**	**99.71**	**93.84**	**99.96**	55.61	23.50
3-layer CNN, $k_{max} = 32$	10,214	**95.55**	83.64	72.85	**97.21**	**99.99**	**96.94**	**99.47**	**97.09**	**99.76**	**97.21**	**99.99**	70.72	47.03
3-layer CNN, $k_{max} = 64$	31,862	**97.15**	84.26	71.72	**98.40**	**99.99**	**98.21**	**99.60**	**98.26**	**99.72**	**98.38**	**99.95**	55.71	14.61
3-layer CNN, $k_{max} = 128$	109,718	**98.36**	86.28	75.44	**98.55**	**99.99**	**98.40**	**99.68**	**98.40**	**99.67**	**98.55**	**99.99**	**97.46**	**97.80**
4-layer CNN, $k_{max} = 128$	245,014	**98.44**	87.63	78.18	**98.52**	**99.97**	**98.36**	**99.65**	**98.39**	**99.70**	**98.53**	**99.99**	**95.25**	**93.43**
5-layer CNN, $k_{max} = 128$	278,870	**98.58**	87.26	77.33	**98.52**	**99.97**	**98.38**	**99.57**	**98.44**	**99.69**	**98.58**	**99.96**	89.88	82.56
6-layer CNN, $k_{max} = 128$	292,002	**98.64**	**94.10**	**90.50**	**98.85**	**99.99**	**98.76**	**99.82**	**98.66**	**99.61**	**98.85**	**99.99**	**98.86**	**100**

Table 4.2: Model ablation study using the CIFAR-10 dataset. k_{max} represents the maximum value of the CNN kernels. We start the analysis from the most straightforward fully-connected linear model. Hidden layers, convolutional layers, or kernel sizes are gradually added or enlarged to test out the most simplistic model that can satisfy an outstanding detection efficiency. We present the training ACC, detection ACC, and BDR for each attack (%); the **boled** results are larger than 90%, which we interpret as satisfying results.

frequency domain can enable more accurate rejection of backdoored data without sacrificing much of the clean samples.

4.3.3 DNN Model Architechures and Ablation Study

Given the different scales of difficulties to separate the DCT data in the frequency domain, we introduce a model ablation study to acquire the most simplistic DNN architecture that satisfies the detection performance to conduct the experiments in Section 4.3.1.

On large-input-spaced samples, namely the PubFig dataset, a linear model would already be able to achieve an outstanding detection efficiency which is introduced in Table 4.1, Section 4.3.2. Thus, no further ablation study is necessary for the large-input-space. The details of the linear model we adopted to conduct the detection task over the PubFig dataset are shown in Table 4.3. We use an Adam optimizer with a learning rate of 0.01 as the optimizer for training this linear model. The binary cross-entropy is adopted as the loss function for the task of linear separation. We train the linear model with 50 epochs on the PubFig based dataset to attain the results shown in Table 4.1.

Given that the DCT results in our evaluation have the same size as the original data's

input space, the DCT results over small-input-space have a weaker ability to depict high-frequency artifacts compared to larger-input-space due to the limited number of high-frequency coefficients. Thus, as shown in Table 4.2, a similar fully connected linear model cannot meet a satisfying detection efficiency over the frequency domain using the same framework we proposed in this work. We then conduct a thorough model ablation study by adding hidden layers or convolutional layers with different kernel sizes to obtain a most simplistic model that meets satisfying detection results over the evaluated attacks as shown in Table 4.2. With more complex architecture and parameters, the DNN can better detect the tested attacks. Based on the ablation study, we found that only until the model's architecture consists of 6 convolutional layers with $k_{max} = 128$ can it meet a satisfying and robust detection efficiency against all evaluated attacks.

Input $(224 \times 224 \times 3)$
Flatten (150528)
Dense (2)

Table 4.3: The network architecture of our simple Linear detector for large input space. We report the size of each layer.

The details of the simple 6-layer CNN detector for the small-input-space are explained in Table 4.4. The above experiments over the small-input-space are evaluated using this model to demonstrate the efficiency of conducting the detection of backdoor triggers in the frequency domain as elaborated in Section 4.3.2. We use Adam with a learning rate of 0.05 as the optimizer to train this model. Other settings are the same as the experiment conducted in large-input-space. The model took 150 epochs over the training set created using CIFAR-10 to converge and attain the results shown in Table 4.1, Section 4.3.2.

Input $(32 \times 32 \times 3)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Max-Pooling 2×2 $(16 \times 16 \times 32)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Max-Pooling 2×2 $(8 \times 8 \times 64)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Max-Pooling 2×2 $(4 \times 4 \times 128)$
Flatten (2048)
Dense (2)

Table 4.4: The network architecture of our simple CNN detector for small-input-space. We report the size of each layer.

4.3.4 Transferability

This section evaluates the transferability of the frequency-based detector towards new datasets. The training set develops the same way as the above experiments. We then test the detector's transferability from a CIFAR-10 model to the GTSRB dataset (Table 4.5). The transferability of a model trained on GTSRB and a model trained on CIFAR-10 to the TSRD dataset (Table 4.6) is also tested.

Table 4.5's column headers indicate the training set used to train the specific detector. For the last column (CIFAR-10+Tune), we first train using the CIFAR-10 dataset, then fine-tune with a 200-sized dataset (half clean, half randomly perturbed originating from the 100 clean samples from the GTSRB test set) of the same distribution as GTSRB. In real life, as the defender is on the user's side, they will have access to the inference data, and a fine-tuning of the model using 100 clean samples is reasonable and practical. Note that the

Attack	GTSRB		CIFAR-10		CIFAR-10+Tune	
	ACC	BDR	ACC	BDR	ACC	BDR
BadNets	90.23	92.55	68.23	99.61	89.44	95.95
Troj-WM	93.96	100	68.42	99.99	91.47	100
Troj-SQ	93.93	99.94	68.40	99.96	91.44	99.95
Nature	91.46	95.00	67.79	98.75	94.03	97.08
Blend	93.67	99.43	66.51	96.18	64.49	45.67
l_2 inv	93.96	100	68.40	99.95	91.45	99.97
l_0 inv	93.93	99.94	68.41	99.98	91.46	99.99

Table 4.5: The transferability using the detector trained on different datasets tested on GTSRB (%).

samples we use to fine-tune the models are not utilized in the test set for all experiments.

Attack	GTSRB		GTSRB+Tune		CIFAR-10		CIFAR-10+Tune	
	ACC	BDR	ACC	BDR	ACC	BDR	ACC	BDR
BadNets	57.99	86.83	77.01	87.10	61.17	98.01	82.53	89.83
Troj-WM	64.57	100	83.46	100	62.16	100	87.10	98.97
Troj-SQ	64.57	100	83.46	100	62.16	99.95	87.58	99.93
Nature	60.09	91.03	83.11	99.29	59.30	94.28	79.61	83.98
Blend	59.04	88.94	82.92	98.92	55.37	86.41	83.62	92.01

Table 4.6: The transferability on the TSRD dataset (%).

When comparing the original GTSRB detector and the CIFAR-10 detector on GTSRB, we see a significant drop in ACC resulting from the variance between the two datasets' data distribution. However, by fine-tuning the detector using the 200-sized dataset, one can achieve a higher ACC without sacrificing too much BDR. The detection efficiency is close to or even surpasses the detector's results with the original GTSRB training set on some attacks. The Blend attack is a particular case here, as the fine-tuned results worsen. We propose the main reason behind this is that the two datasets are of significant variance in distributions. This side effect over the detection deficiency against Blend is recovered in the following experiments using pairs of training and testing sets with closer distributions.

Table 4.6 presents the results on evaluating the detector's transferability from the GTSRB and CIFAR-10 datasets onto the TSRD dataset. Due to the limited size of the TSRD

dataset, we cannot achieve satisfying accuracy using the target model presented above; therefore, the TSRD dataset is only for testing. The raw detector results are similar to the experiment evaluating the CIFAR-10 model over GTSRB test data. After fine-tuning with 100 TSRD clean samples (dataset of size 200), both detectors can achieve satisfying detection results with acceptable ACC on the TSRD dataset. Of note, after fine-tuning, both detectors' performances against the Blend attack over the TSRD dataset are better than the results from the previous experiment's over the GTSRB dataset. We believe this is because of closer similarities in distribution between the datasets than between CIFAR-10 and GTSRB.

	Combined		Combined+Tune	
Attack	ACC	BDR	ACC	BDR
BadNets	64.28	89.88	80.28	89.80
Troj-WM	69.34	100	85.28	99.80
Troj-SQ	69.34	100	85.36	99.95
Nature	64.67	90.66	83.29	95.82
Blend	64.18	89.68	84.61	98.45

Table 4.7: The transferability with extended training set, tested using the TSRD dataset (%).

We also notice the CIFAR-10 detector achieves higher accuracy than the GTSRB detector on the TSRD dataset for most cases, even though CIFAR-10 and TSRD have disparent sample categories. Given that the two detectors are all trained with the same number of epochs and settings, we deduce that the transferability is related to the training set's size. This assumption is confirmed in the following experiment when evaluating the transferability using a combined training set of CIFAR-10 and GTSRB. As shown in Table 4.7, when using a combined dataset, we can see an improvement in the average detection efficiency over the TSRD dataset.

Remark 2. *Since the high-frequency artifacts of existing triggers are universal across different datasets, transfer learning can be adopted in the task of detecting backdoored samples in the frequency domain. Even if the defender does not have access to the original training set, they can still effectively detect attacks and achieve satisfying results in the frequency domain by adopting large public clean datasets to conduct transfer learning.*

4.4 Creating Smooth Triggers

4.4.1 Problem Definition

Given existing attacks' high-frequency artifacts, this section aims to create triggers that do not leave high-frequency artifacts but stay efficient as backdoor triggers. We summarize generating a smooth trigger as a bilevel optimization problem:

$$\min_{\delta} \Sigma_i L(x_i + \delta, y_{tar}; \theta_p) + \lambda \Omega(\delta; g), \tag{4.1}$$

$$s.t. \quad \underbrace{x_i + \delta}_{i=1,\dots,N} \in [0,1]^n, \tag{4.2}$$

$$\theta_p = \text{argmin}_\theta \left(\Sigma_i L(x_i, y_i; \theta) + \Sigma_j L(x_j + \delta, y_{tar}; \theta)\right) \tag{4.3}$$

We adopt $\Omega(\cdot; g)$ from SmoothFool [35] to measure the input sample's roughness given a preset low-pass filter in the image domain g. λ is the Lagrangian coefficient that controls the trade-off between smoothness and perturbation scale. Equation (4.1) is the optimization problem that tries to minimize both the loss of the poisoned data given a trained poisoned model and the roughness of the trigger itself. Equation (4.2) ensures the poisoned samples falls within the rational range from $[0, 1]$. Equation (4.3) is the optimization problem to train a poisoned model where θ_p is the poisoned model, and θ is an initialized target model, where j is the index of the samples selected as poison data.

4.4.2 Methodology

There are two ways to achieve the constraint of smoothness with the low-pass filter. One way is to conduct the search iteratively and output the results when it meets the constraint. However, we find this methodology is ineffective in our case as optimization along the gradient of DNNs causes local "bumps" in the triggers that easily exceed the constraint. Therefore, we adopt a strategy by updating the smooth trigger with the perturbation that remains after the low-pass filter for each iteration, thus meeting the constraint. The remaining parts of the perturbation from the filter can be interpreted as $r = \delta * g$. Here, r is the result of the perturbation after convolving with the low-pass filter, g, in the image domain. Taking Equation (4.2) into account and the fact that the triggers are of small values after passing through g, we adopt a min-max scaler, M, as a normalization process to remap the poison data onto the rational range of an image, $[0, 1]$. Instead of using the rigid value clipping done in other works, we argue that normalization can better keep the relative scale between each pixel of the smooth trigger and better maintain functionality as a backdoor trigger. Consequently, we can rewrite the optimization as:

$$\min_{r} \Sigma_i L(x_i^{poi}, y_{tar}; \theta_{poi}), \tag{4.4}$$

$$s.t.\ \ r = \delta * g, \tag{4.5}$$

$$\underbrace{x_i^{poi} = M(x_i + \lambda r)}_{i=1,\ldots,N}, \tag{4.6}$$

$$\theta_{poi} = \text{argmin}_\theta \left(\Sigma_i L(x_i, y_i; \theta) + \Sigma_j L(x_j^{poi}, y_{tar}; \theta) \right) \tag{4.7}$$

This bilevel optimization function's objective is to find a smooth pattern r within the range of the low-pass filter g that can be successfully adopted as a backdoor trigger. As stated in our work's scope, the classifier θ is a DNN, thus making the optimization problem non-convex [117]. Thus, we propose Algorithm 3 to approximate a solution to this problem:

we heuristically search for a smooth pattern that leads clean samples to the target label.

Algorithm 3: Generating a Smooth Trigger

Input: Data Points: $X \in R^{N \times H \times W \times C}$;
Pre-trained Classifier: θ;
Desired Fooling Rate: γ;
Output: Smooth Trigger: r; Dominante Label: y_{tar};
Parameters: Low-pass Filter g; Trade-off Controller: λ; Number of Classes: K

```
/* Initialization */
```
$r \leftarrow 0^{H \times W \times C}$;
$y_{tar} \leftarrow randint(K)$;
$\gamma^{best} \leftarrow Err(X)$;
while $\gamma^{best} < \gamma$ **do**
  for *each data point* $x_i \in X$ **do**
    if $\theta(M(x_i + \lambda r))! = y_{tar}$ **then**
      `/* Computing Purturbation */`
      $\delta = -\nabla L(x_i, y_{tar}; \theta)$;
      `/* Low-Pass Filter */`
      $r = r + \delta * g$;
      $r = r * g$;
  $X_{poi} = M(subset(X) + r)$;
  $y_{tar} = Domi(X_{poi})$;
  if $Err(X_{poi}) > \gamma^{best}$ **then**
    `/* Updating the Best Result */`
    $\gamma^{best} \leftarrow Err(X_{poi})$;
    $r^{best} \leftarrow r$;
    $y_{tar}^{best} \leftarrow y_{tar}$;
return r^{best}, y_{tar}^{best}

Algorithm 3 explains the procedure of generating a smooth trigger. $Err(\cdot)$ computes the error rate, and $Domi(\cdot)$ outputs the mode of the labels that are different from their original ones. The algorithm first initializes a random target label and a zero-image as the trigger. While the error caused by the generated trigger is below the desired fool rate γ, the algorithm will iteratively compute the perturbation according to the gradients of a pre-trained model towards the target class for each sample that is not of the target label. The attained perturbation then passes through a low-pass filter to remove high-frequency parts.

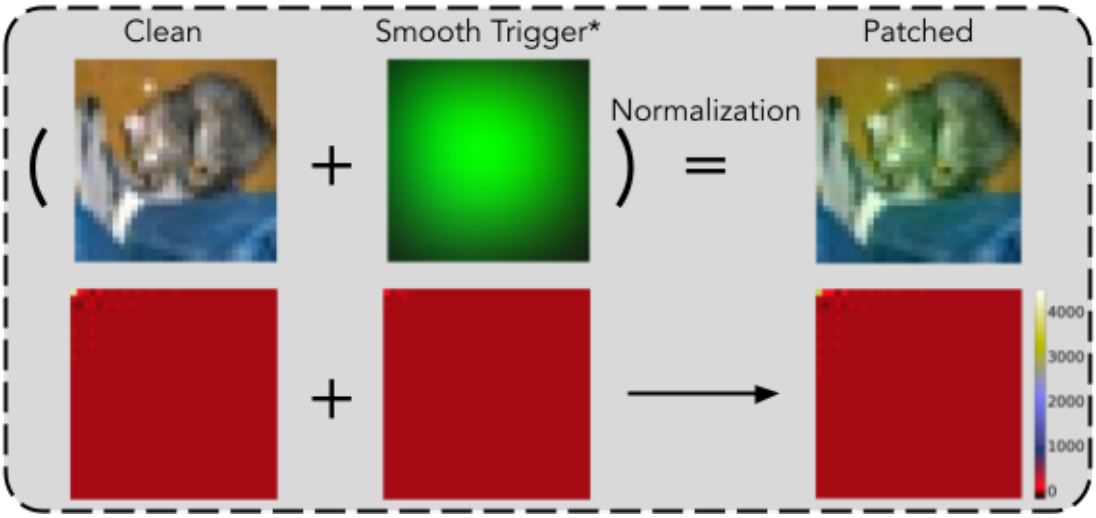


Figure 4.9: Visual effects over image and frequency domian of the smooth triggers. The trigger is multiplied by 5 for visualization. The right bottom depicts the heatmap averaged over 10000 samples patched with the smooth trigger. Both the trigger itself and the final images exhibit frequency spectra similar to natural images.

The smoothed perturbation is added to the trigger to update the smooth trigger. Finally, we select out a subset from all the data points to quickly estimate the new error rate. If the estimated error rate is larger than the preset threshold, we will update the best smooth trigger pairing with the dominant label.

Upon experiments of generating a unified perturbation aiming to cause universal misclassification [117], there exist several dominant labels that perturbations tend to lead to. We compute the dominant label as the target label and pair it with the corresponding smooth trigger to achieve a more potent backdoor attack.

4.4.3 Attack Results and Evaluations

Figure 4.9 depicts the computed smooth trigger's visual effects using the proposed algorithm in the image domain and frequency domain. A similar figure illustrating the smooth trigger generated based on the GTSRB dataset is presented in Figure 4.10. As one can see

Figure 4.10: Visual effects over image and frequency domian of the smooth triggers. The trigger is multiplied by 5 for visualization. The right bottom depicts the heatmap averaged over 10000 samples patched with the smooth trigger. Both the trigger itself and the final images exhibit frequency spectra similar to natural images.

from the frequency results, neither the trigger itself nor the final patched image contain any high-frequency components. Meanwhile, the instances patched with the triggers still keep the original information as the main body, as shown in the upper left of Figure 4.9 and Figure 4.10. Based on the comparison from both the image and frequency domain, we can say the smooth trigger has better latency than all evaluated triggers.

We now evaluate the smooth trigger's functionality as a backdoor trigger by using it to poison the training set and conduct the entire backdoor attack pipeline. We adopt a small CNN trained on CIFAR-10 with an ACC of 85.50% as the baseline model. Table 4.8 shows the details of the pretrained model. The model was trained using Adam optimizer with a learning rate at 0.05 for 150 epochs to converge. The base-line ACC over clean samples is 85.50% for the CIFAR-10 dataset. We also trained a base-line model on the GTSRB dataset for generating the smooth trigger over the GTSRB dataset. The GTSRB base-line model's ACC is 97.45%.

The smooth attack can attain an *Attack Success Rate* (ASR) around 95% within one epoch of training while the model's training accuracy is still below 30%. This effect indicates the smooth trigger contains features that are easier to pick up by the DNN. We evaluate the final result when the model converges over the poison dataset with a poison

Input $(32 \times 32 \times 3)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Conv2d 3×3 $(32 \times 32 \times 32)$
Max-Pooling 2×2 $(16 \times 16 \times 32)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Conv2d 3×3 $(16 \times 16 \times 64)$
Max-Pooling 2×2 $(8 \times 8 \times 64)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Conv2d 3×3 $(8 \times 8 \times 128)$
Max-Pooling 2×2 $(4 \times 4 \times 128)$
Flatten (2048)
Dense (10)

Table 4.8: The target model for evaluating the smooth trigger on CIFAR-10 and GTSRB dataset. We report the size of each layer.

ratio of 0.1, which is a standard poison rate used in other attack works [94, 104]. The poisoned model recognizes the trigger by 97.25% of chance and achieves an ACC on clean samples at 84.54%, which is close to the baseline ACC.

As a comparison, we test the case of using random patches and nature images passed through the low-pass filter as naive designs of the smooth triggers. The triggers can only reach an average ASR of 75.54%. Meanwhile, we observe that the naive-designed smooth triggers take more epochs for the model to converge. The averaging ACC over clean samples can only achieve 76.29%, with five naive-designed smooth triggers considered. This drop in the performance over the clean samples can also impair the stealthiness of the attack.

We show a similar pattern on the GTSRB dataset. The model trained over the poisoned GTSRB dataset can maintain an ACC over clean samples at 97.42%, which is almost the same as the base-line model. Meanwhile, the ASR is 97.86% without defense. We observed the model could achieve an ASR greater than 90% even with one epoch of training.

Thus, we conclude that our smooth trigger maintains functionality as a backdoor trigger while leaving no high-frequency artifacts.

Remark 3. *Directly using random patches passed through the low-pass filter cannot generate smooth triggers of satisfying functionality. We show that by approximately solving a bilevel problem, one can generate smooth triggers that function as backdoor triggers while achieving a satisfying stealthiness in both image and frequency domains.*

4.4.4 Impacts over Defenses

To show the importance of considering smooth triggers in defenses, we perform a small case study on *Meta Neural Analysis* (MNA) [20], a state-of-art defense mechanism. When faced with a classifier poisoned with a smooth trigger, the MNA can only achieve an AUC score of 0.0776. However, after upgrading the MNA to consider the smooth trigger generation, the upgraded MNA can achieve an AUC score of 0.694 and a detection accuracy of 42.85%. This simple case study illustrates how existing defenses can be made more robust by considering smooth triggers.

Similarly, we also aim to upgrade our proposed detector with smooth triggers. We first try to finetune the detector with samples patched with patterns passed through the low-pass filter. Although the detector successfully detects samples patched with the same trigger with 95.67% accuracy, the detector fails to generalize and cannot detect other filtered triggers nor the smooth trigger. We next experiment using the smooth trigger we acquired using Algorithm 3 to finetune the model for one epoch with 20,000 samples (half clean, half patched). This time, we find the model performs well on detecting the smooth trigger (82.49% accuracy) and attains a higher detection rate of 89.37% averaged over all unseen low-pass filtered triggers. With this detection rate, the detector can constrain the overall ASR of the most potent smooth trigger found using Algorithm 3 to 19.72%. If we can use

the detector to eliminate poisoned samples in the training set, we further drop the overall ASR to 18.03%.

Again, we see a similar story with the GTSRB dataset. The detection rate of the proposed detector in Section 4.3.1 can only achieve a BDR at 55.31% and an F1 score at 0.664 before considering this smooth attack. This detection efficiency can only drop the attack success rate of this GTSRB smooth trigger to 40.97% By incorporating this strongest smooth trigger found using Algorithm 3 into the development of the detector, we can regain a high efficient detection efficiency of a BDR at 85.53% and an F1 score of 0.8628 using the fine-tuning pipeline proposed in Section 4.4.4. This fine-tuning does not affect much over the other attack trigger's detection efficiency due to the limited scale as discussed in Section 4.4.4. Using this upgraded detector on the poisoned model, we can finally constrain the ASR from 97.86% to 13.27% by only adopting the detector to reject samples with triggers during the inference. In the case where we apply the detector to the training phase , we can further drop the ASR to 13.03%.

We design an experiment comparing the distance in the hyperplane between clean samples and samples patched with filtered triggers (including the smooth trigger and other simple designs) to better explain this generalizability. We take the detector's last layer's weight on the benign class and compu te the Euclidean distance between the weights and the clean samples' logits to select the "representative" of the clean cluster in the hyperplane. We then feed the poisoned samples patched with different kinds of low-pass filter processed triggers to acquire the average distance between the clean representative and the poisoned samples' clusters. We find that the smooth trigger patch samples have the closest distance of 4.3589 among all the filtered triggers. For reference, low-passed Blend is 4.6040; low-passed Nature is 4.5561; and random noise passing through the low-pass filter has a distance of 4.6036. Figure 4.11 helps explain the generalizability acquired by fine-tuning the detector using the smooth trigger. With a closer distance toward the clean sample

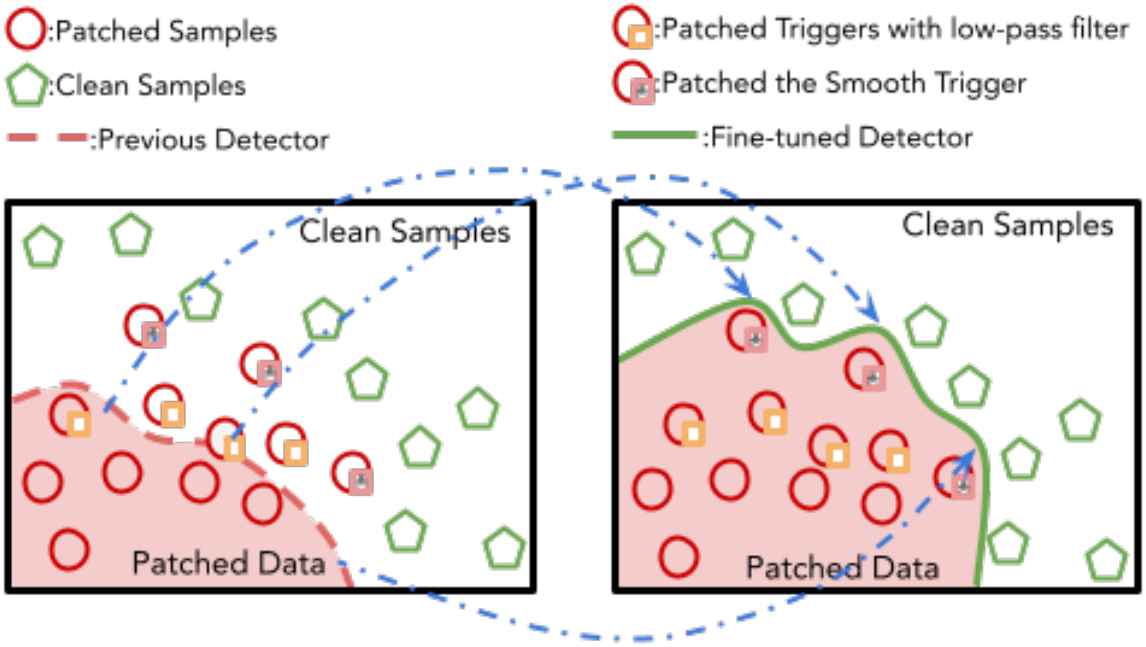


Figure 4.11: Fine-tuning over the smooth trigger patched samples

center, the smooth trigger-patched samples can work as support vectors in the hyperplane to include other filtered triggers, thus achieving universal generalizability.

Remark 4. *We show that defenses designed with the frequency domain considered can better mitigating the smooth triggers. We bring attention to the development of frequency-constraint triggers, as they can be adopted in an adversarial training manner to help defenses acquire robust and generalized protection against smooth triggers.*

4.5 Conclusion

In this work, we filled the gap in existing works on backdoor attacks and defenses by presenting a comprehensive analysis of the overlooked frequency domain. Unlike natural images, we found many existing attack triggers exhibit severe artifacts in the high-frequency spectrum. We took advantage of the artifacts and show that we can achieve an average detection rate of 98.50% under attack-agnostic settings. Realizing this limitation in the

current trigger design, we proposed an effective way to generate triggers invisible in the high-frequency domain. We demonstrated its potency in terms of stealthiness and attack efficiency. Finally, we showed that existing backdoor defenses could benefit from considering frequency-invisible attacks. We hope the remarks and solutions proposed in this chapter can inspire more advanced studies on backdoor attacks in the future.

CHAPTER 5

Detection of Security Anomalies in Industry

5.1 Introduction

A common defense against attacks is to detect anomalies to inputs before they occur, similar to how a firewall will inspect and sanitize incoming web traffic before allowing passage into an internal network.

The task of anomaly detection involves aiming to identify points, events, and/or observations that deviate from a dataset's normal behavior. Anomalous data can indicate critical incidents, such as a technical glitch, or potential opportunities, for instance a change in consumer behavior. Anomaly detection is important in various applications including networking and telecommunication server monitoring

In fact, a common problem faced by companies is detection of anomalous events in time series in which the data consists of a sequence of values over time. Such a task might be seen, for example, in identifying health in telecommunication networks, tracking performance of applications and infrastructure components, and monitoring key performance indicators (KPIs). However, anomaly detection is often difficult as the anomalous behaviors are difficult to detect due to problems like large dimensions caused by multiple values for the same time stamp (multivariate time series) or potentially benign noises in the data. These situations are ever-present in large-scale data faced by companies that have added

time and memory constraints. This is especially true at an industry environment such as Ericsson which sees gigabytes if not terabytes of data every day. To make matters more complicated, any existing system needs to be cognizant of the practical constraints in terms of time and cost requirements - a system cannot take too much time or memory or else it will not be useful.

Currently there exist two common techniques widely employed by corporations for anomaly detection large-scale data: Density-based spatial clustering of applications with noise (DBSCAN) and median absolute deviation (MAD) [139]. [1]

DBSCAN is a density-based clustering algorithm that greedily groups together points that are within ϵ distance of one another. A minimum of n points have to be within ϵ distance of one another before the points can be considered a cluster. Any point not in a cluster or clusters that have relatively few points in them are considered outliers.

Median absolute deviation is a measure of dispersion designed to be more robust than standard deviation. Given data points $X_1, X_2, ..., X_n$ with median $\tilde{X}$, the MAD is defined as the median of the absolute deviations from the median:

$$\text{MAD} = \text{median}(|X_i - \tilde{X}|)$$

To apply the DBSCAN and MAD algorithms to time-series data, we start with the time series points we want to test. We then obtain data in the past that should theoretically be most similar to this data (e.g. data from the same time in the previous week or month). DBSCAN and MAD is then run comparing the historical data and the test data to determine if the latter is an anomaly using a predetermined threshold.

These two techniques have been the standard practice as they are appealing for a number of reasons. First, they do not take very much time to run, which is critical when the

[1] Even though we cite this blog, we are not singling out this company. In fact DBSCAN and MAD are commonly employed strategies in many areas.

algorithm would possible be needed to be run on large amounts of data. Secondly, because there are not many hyperparameters, the algorithms are also quick to tune. Finally, these algorithms are also easy to interpret and comprehend - it is intuitive what the outputs of these algorithms are and why the results come out as they do.

However, there are disadvantages to these standard approaches. One glaring one is that MAD works well only on *univariate* time series data and can only one KPI metric at a time. Not only could this be more time-consuming the more KPI values there are, but valuable information that could be contained in the relationship *between* KPI values is lost. This is the case even for DBSCAN, which can be modified to run on multivariate time series. Another downside is that these methods inherently assume a proper seasonality in the data. If none exist or if the seasonality is not strong enough, DBSCAN and MAD would fail to provide accurate results.

DBSCAN and MAD also have a large memory requirement since they require the storage of all the data in readily accessible memory. The more accurate the desired result, the more historical data would need to be used and the more memory this would require. In short, there is a direct growth of memory requirement needed to maintain a level of accuracy, which poses challenges to their scalability.

To address these challenges, we propose a machine learning based approach called ERICA for anomaly detection on time series data. The list of contributions for this work are as follows:

- We propose an autoencoder based deep learning model and a corresponding custom pipeline with pre-processing and post-processing steps called ERICA.
- We then evaluate on real world datasets to show that our technique is more resilient to seasonality disruptions and results in a higher accuracy

In this work, we propose our autoencoder based deep learning approach in section 5.2.

The evaluation result is presented in Section 5.3. Finally we conclude and discuss future work in Section 5.4.

5.2 Our Approach

5.2.1 Considerations of Other Approaches

As mentioned in Chapter 2.3, we wanted a solution that was powerful enough to handle the raw amount of data and be robust to minor noises. At the same time, we had to be mindful of resource constraints and avoid resource-heavy tools like generative adversarial networks (GANs).

We decide to explore an approach in the middle ground for our machine learning architecture: a solution that is complicated enough to handle multivariate, potentially complicated time series data. However, we also need to keep in mind that the technique should be relatively simple to implement and understand as well as be mindful of time and storage requirements. There are two areas that we derive our ideas from.

The first involve using RNNs [197] or LSTMs [67] to capture the features in the time series data. . Because we are trying to identify patterns in time-series, we decided to utilize Long Short Term Memory (LSTM) units to incorporate historical pattern and data in the learning process. LSTMs are designed to capture / learn representations for sequences.

Simplified, LSTMs use cells to capture the encoding at every time step. Each cell uses the encoding from the previous cell, which represents the previous time step. However, the key difference in LSTMs is that each cell contains ”gates” which allow cells to ignore or ”forget” inputs from the previous cell. This allows the model to capture long-term dependencies and in the case of our framework, long-term seasonal patterns. Time-series data naturally contains a sequential dependency in which an event that occurs at a certain time

is likely to affect an event later. Utilizing an LSTM helps improve model performance and capture anomalies and patterns that depend on previous events.

The second line of work that we took inspiration from involve the use of an autoencoder based architecture, which are commonly used to learn efficient codings of unlabeled data [3, 187, 159], taking advantage of the fact that the autoencoder architecture with its encoding and decoding phases lends very well to the task anomaly detection. At a high-level, the autoencoder architecture forces the model to learn a compressed knowledge representation of the input. This is done by creating a ”bottleneck” in the network. Before the bottleneck, the autoencoder layers (called the encoder) attempts to learn a simplified representation with fewer dimensions (called the latent representation or encoding). We will call this encoding $\mathbf{e}$. The number of dimensions needs to be carefully chosen to balance oversimplifying the data and losing information versus not simplifying enough and reducing the effectiveness of the autoencoder. The layers after the bottleneck, called the decoding layers or the decoder, ”tests” the quality of the latent representation by attempting to reconstruct the original input. The model then measures the difference between the original input and the reconstructed input. Since we assume that almost all of the training data will be non-anomalous data, the autoencoder is able to capture the ”normal” pattern while training. During inference time, any anomalous pattern will be unable to be reconstructed correctly and result in a higher error and could be identified.

An added benefit is that the act of dimensionaltiy reduction is inherently built in into the architecture,which allows us to use the appropriate level of abstraction without losing the granularity needed to represent the feature space.

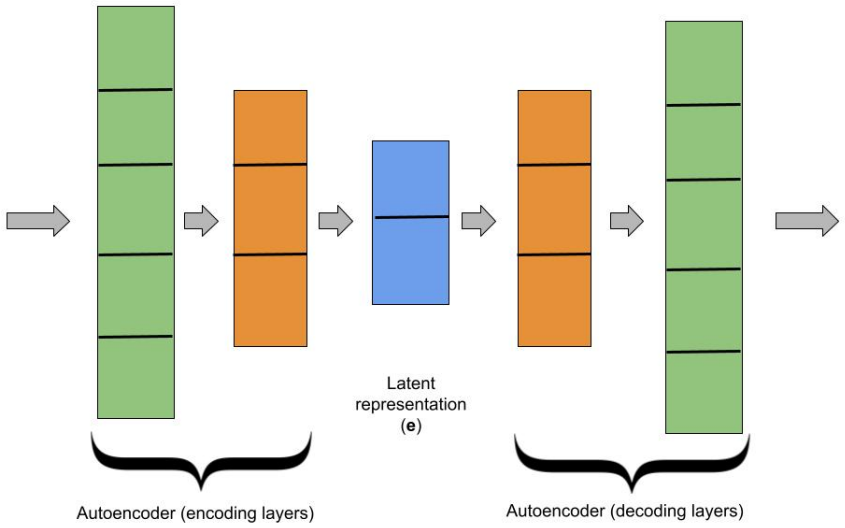


Figure 5.1: High level depiction of general autoencoder architecture

5.2.2 Model Architecture

We propose an autoencoder based deep learning framework for anomaly detection, the reasoning for which is explained in Chapter 2.3.

In short, the final architecture we decided on is an autoencoder-based architecture combined with an LSTM. The input to the model is a multivariate time-series data that has been pre-processed. The details of the pre-processing step are laid out in Section 5.2.4. The model outputs a value for every time instance inputted into the model, which represents the reconstruction probability from **e** - in other words, it roughly measures how likely the input is from the stored parameters. Very unlikely probabilities means the input is likely anomalous.

We use a specific type of an autoencoder called a variational autoencoder (VAE), commonly used in representation learning. The main difference is that in a VAE we assume that the input data can be characterized by an unknown probability distribution which we aim

to model. The encoder and decoder are trained jointly to minimize the divergence between the true posterior and the parametric posterior. The latent representation is stored as the mean and variance of a Gaussian distribution instead of a singular value. The layers after the bottleneck - called the decoder - attempts to recreate the original input from the latent space. The effectiveness of the model is measured by the difference between the original input and the recreated output. We choose to use a VAE in short because Bayseian models like VAEs give more control over how to model the latent distribution, something that is no possible in the vanilla autoencoder framework. Such a control is important because different time-series data are likely to have difference in their underlying distribution patterns from one another.

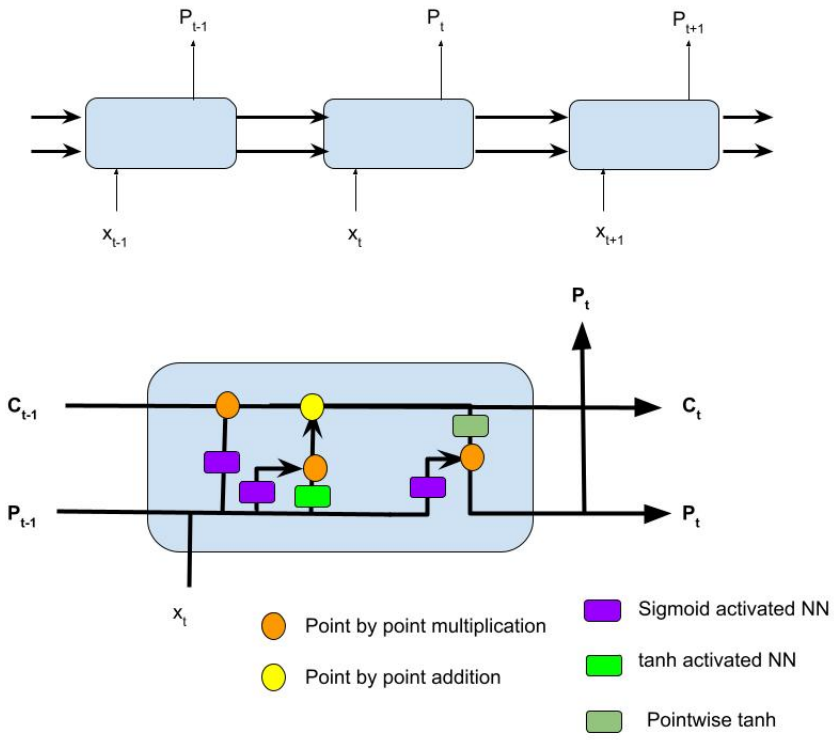


Figure 5.2: High level depiction of LSTM units. x represents the input, c represents the cell state vector, which controls which values to "forget", while P represents the hidden state or output vector of the LSTM

5.2.3 Model Details

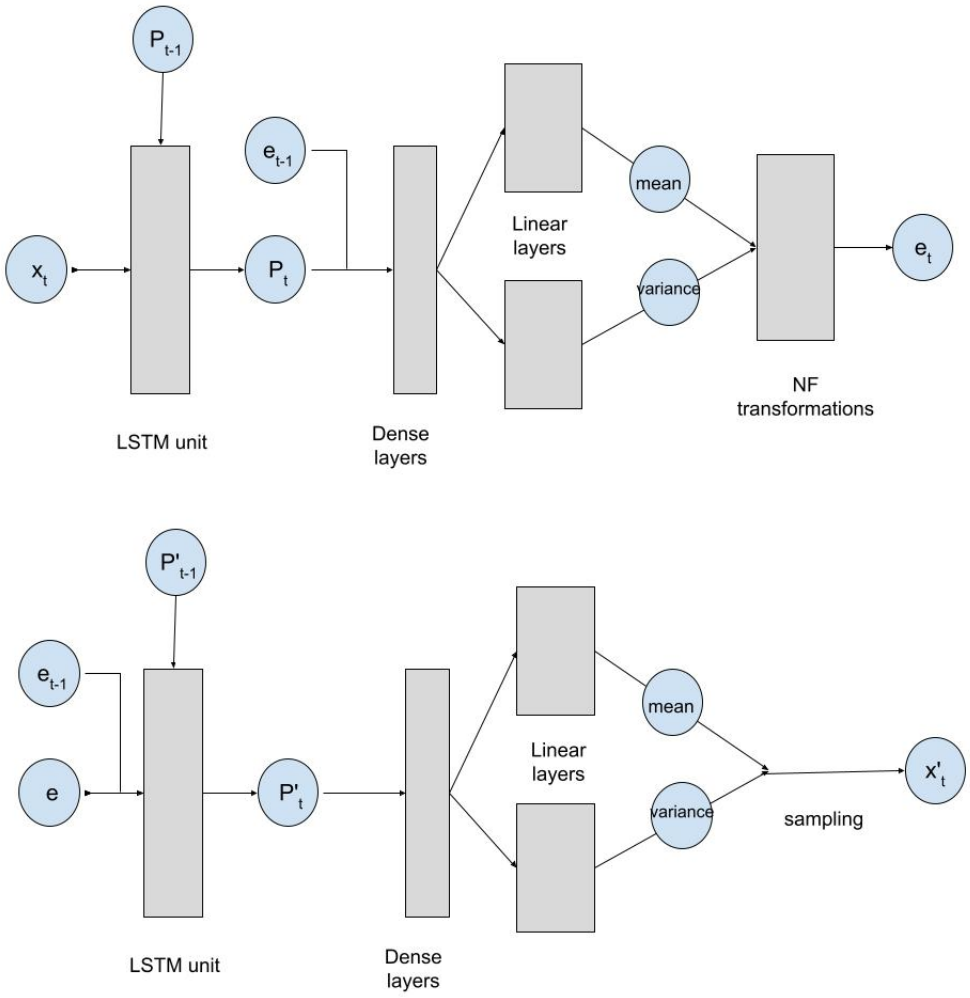


Figure 5.3: Detailed model architecture of encoder layers (top) and decoder layers (bottom)

The following section delves into more details for the encoder and decoder networks. For each section, assume that an input corresponding to KPI values at time t is inputted (marked c_t in the diagram).

Encoder Network: The first part of the model is the LSTM unit which takes in the input data of dimensions (batch size x window size x number of KPIs). It then outputs a vector

of size (batch size x window size x number of hidden RNN cells). We will call this vector P_t. The number of hidden RNN cells is a hyperparameter of the model that is set before the model is run.

The vector P_t is combined with the latent vector from the previous time step (e_{t-1}) and fed into linear layers. These layers perform non-linear transformations to obtain the variance and mean. After sampling from a Gaussian distribution with the given variance and mean, and undergoing planar transformations, the final output is the encoding for the current time step (e_t).

Decoder Network: The input to the decoder network is the latent vector for a given time step (e_t), which is fed into an LSTM as above. The output, which we call P' is also a vector of size (batch size x window size x number of hidden RNN cells). P' is fed into linear layers to obtain the bias and variance of a Gaussian distribution. The final reconstructed input x' is sampled from this distribution and the output is the reconstruction loss (r):

$$r = logP_\theta(x_t|e_t)$$

where x_t represents the input at time t and e_t represents the latent variable at time t.

5.2.4 Proposed Pipeline

Our proposed pipeline for training and supporting ERICA consists of several steps: pre-processing, model training, threshold selection, model evaluation, and post-processing. An overview can be seen in Figure 5.4.

1) Pre-processing: The first step in our pipeline is data pre-processing. This step is done on both training data – historical data that we use to create the model - and evaluation data –

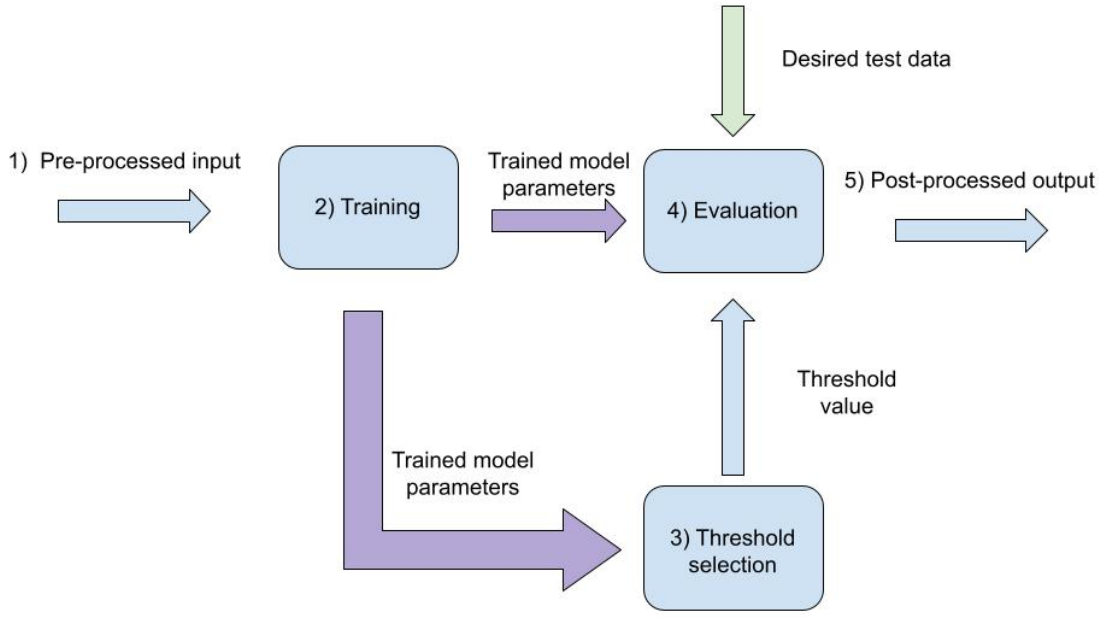


Figure 5.4: High level overview of pipeline

the data that we wish to test. The input data are performance metrics (or KPI values) taken at increments over a specific time. In this stage, we convert all metrics to the corresponding KPI, if it is not already in that form. If any extraneous KPIs are present, they would be removed in this step. Next, any data conversion will be applied to convert to stationary seasonality. After all this, normalization occurs in which we convert the range of values to between –1 and 1. Finally, the values are converted into “sliding windows” - a predetermined number of consecutive values are grouped together. The ideal value for the number of time steps in a window is decided ahead of time. Ideally it should be large enough to capture enough ”patterns” in the data and would vary based on the particular data used.

2) Model training: With this pre-processed data, the model is trained offline. After training, the result is a file of model weights, which define all the parameters of the model. These weights are saved in ready to access memory. Unlike in DBSCAN and MAD, there is no

need to store the training data except for archival purposes.

3) Threshold selection: In addition to the model, a threshold also needs to be set. The purpose of the threshold values is to distinguish the value below which the outputs would be considered an anomaly. The threshold selection is done once after training but can also be done throughout the life-cycle of the model to update the value. In either scenario, the process is similar. First, the model is evaluated with some pre-processed data. This can be done with the data used to train the model or a mixture of training data and new data. The threshold value can be set via a simple percentile (e.g. 98.9) of the resulting output or utilizing peaks-over-threshold models[88].

4) Model evaluation: This step involves taking the trained model weights and running it on the unseen data to test for anomalies. After pre-processing, the desired unseen test data is fed into the model resulting in anomaly likelihood scores. Once evaluation is complete, the test data no longer needs to be kept in memory.

5) Post-processing: After the likelihood scores are obtained, some post-processing steps are taken. Firstly, for any anomalous time index (i.e. output value below threshold value), we check to see if the time instances before and after the time index were also flagged. The reasoning for this is twofold: 1) anomalous events usually have effects lasting multiple time stamps and 2) because of the sliding window approach, the same anomalous timestamp would occur in multiple sliding windows. This process helps filter out potential false positives. Then, any output values above the predetermined threshold are discarded. Finally, amongst the remaining values, consecutive anomalies are discarded. For example, if an anomaly occurs at time t, any anomaly happening with the next x minutes (where x is a predetermined value) are also discarded. This helps prevent an overload of alerts that

	DBSCAN			MAD			ERICA		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SwAT	0.63	0.55	0.59	0.76	0.67	0.71	0.89	0.74	0.81
WADI	0.52	0.49	0.51	0.55	0.54	0.54	0.59	0.61	0.60
CICIDS 2017	0.93	0.88	0.91	0.92	0.89	0.90	0.94	0.92	0.93
UNSW-NB15	0.76	0.74	0.75	0.78	0.76	0.77	0.86	0.84	0.85

Table 5.1: Best F1 scores for various approaches on the datasets

correspond to the same fault. The post-processing step also allows for human feedback to adjust thresholds and blacklist alerts as needed.

5.3 Evaluation

In this section, we compare our machine learning model against the popular established techniques of DBSCAN and MAD since we aim to draw attention to the effectivness of ERICA over the aformentioned traditional methods. Due to the sensitivity of potential confidential network data, we evaluate on real world datasets: SwAT, WADI, CICIDS 2017 [151], and UNSW-NB15 [121].

5.3.1 Datasets

SwAT and WADI are both public datasets related to monitor water related systems [70], with 51 and 123 columns respectively and containing about 4.65% and 5.76% anomalous events respectively, each lasting for at least a minute. The SwAT training data contains data collected over a week (496800 time stamps) and the evaluation data contains data collected over a period of 5 days (449919). The training data for WADI contains data collected over 15 days (129605 points) and the test data contains data collected over 2 days (172800 time stamps).

CICDS 2017 consists of normal and attack traffic data collected over a five-day period.

For training, the normal (non-attack) time series samples were used. UNSW-NB15 is a network intrusion dataset, which contains nine different attacks. As similar to CICDS2017, training was done on the non-attack samples.

5.3.2 Configuration

For DBSCAN and MAD, we assume the best case scenario and we set all tunable values (e.g. ϵ for DBSCAN and the threshold for MAD) such that there are no false positives when run on the full training data. This was done via grid-search for every feature. We then run the algorithm on the evaluation data and record the best scores in the table.

For ERICA, we train on 10 epochs on the full training data. We set the dimension of the latent space to 25 and the number of dense layers to 500. A simple ablation study of these figures can be seen in Figure 5.5.

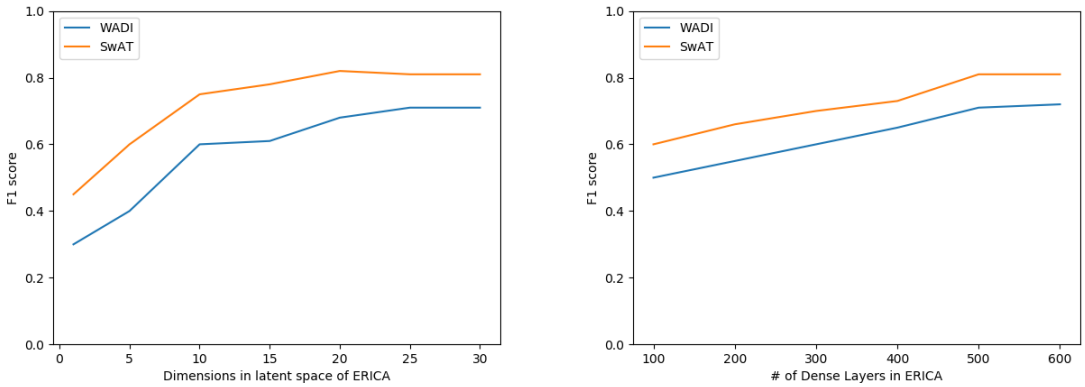
Training was performed on GPUs and took on the order of tens of GPU hours. While this is to be expected for a relatively complicated model with a large amount of data, it is imperative in real world applications that training be performed offline. However, evaluation on a single window can be performed quickly, at most requiring a few seconds. In fact, once a time series data has been collected, it can immediately be fed into the pipeline for anomaly detection for quick turnaround, This is in contrast to DBSCAN and MAD which takes over a minute to run on all the features for all the training data.

5.3.3 Results

When evaluating on the test data, we run the post-processing step and deem a window as a detected anomaly if any post-process data signals an alert in that window. Using this, we report the best F1 score for the different methods. We find that DBSCAN and MAD are strongly affected by noise and are unable to reach high levels of accuracy. The full results

can be seen in Table 5.1. Due to the stochastic nature of the VAE, we run the model for 5 times and report the average of the instances

The experiment also makes apparent the potential memory savings - while our saved model takes roughly 25 MB in memory, the saved historical data occupies at least 120 MBs. Keeping in mind this is a relatively small dataset, the disparity is only likely to increase when faced with real world data.



(a) Ablation study of latent space

(b) Ablation study of dense layers

Figure 5.5: Our ablation study showing our model's best F1 score against changes in number of dimensions in the latent space (left) and number of dense layers (right)

5.4 Conclusion and Future Work

In our work, we highlight potential shortcomings of current anomaly detection methods deployed in industry. We propose a machine learning framework based on an autoencoder based architecture and show that on complicated datasets, our proposed framework can surpass existing techniques. We encourage future work to explore this area further and to build more robust and reliable systems to deal with the problems encountered in real world industry applications.

CHAPTER 6

Adversarial Fine Tuning

6.1 Introduction

Ever since the discovery of adversarial examples [164, 48], there have been lots of study on defenses to help mitigate it. As mentioned in Chapter 2, much of the defenses have shown to be ad-hoc and fail in the presence of adaptive attacks - attacks that know the defense exists.

That has left two main lines of defenses that have stood the test of time. The ideal solution is to utilize defenses called *certified robustness*, which provide a *provable* guarantee that the model will not misclassify an input given that the distortion is within some bounds. Some of these works include [181, 113, 49, 32, 31, 89, 92]. However, while stunning progress has been made on this front, the bounds that are guaranteed are relatively small, and not yet useful in a practical, real-world setting.

So far, the most practical defense we have is an empirical based defense, a popular one being adversarial training [48, 109]. The goal of adversarial training is to train a model from scratch to not only perform well on the unaltered, benign data but also perform well under adversarial attacks. The ending result is a model that *usually* performs slightly worse on unaltered, benign inputs (benign accuracy) as compare to its vanilla model counterpart, but makes up for it by performing substantially better on adversarial inputs.

More formally, in the case of a normal classifier, the user tries to minimize some objective function L which takes in some model F with parameters θ, input x, and output y:

$$\min_{\theta} L(F_\theta(x), y)$$

Using the above formulation, we can state that an adversary, with the goal of creating an adversarial example tries to solve the following optimization problem:

$$\max_{\delta \in \Delta(x)} L(F_\theta(x + \delta), y)$$

In this case, δ is the perturbation added to an input x and we bound the δ by some bound $\Delta(x)$ that may be dependent on the input (e.g. a L_2 norm-ball). The adversary is trying to maximize the error loss between the original input x and the original output y. Note that this is for an *untargeted* adversarial attack though the loss function can be modified slightly for a targeted adversarial attack and the same principles hold.

If we combine the two equations, in order to build a robust model against advserarial examples, we can see that we are trying to minimize the *expected* or empirical loss caused by an adversarial attack for a any model.

$$\min_{\theta} \frac{1}{|S|} \sum_{(x,y) \in S} \max_{\delta \in \Delta(x)} L(F_\theta(x + \delta), y)$$

In the above equation S is the training set.

As an oversimplification, adversarial training solves an adversarial minimax problem in which in each epoch of training, the ”adversary” tries to maximize the training error with a malicious input (adversarial example) and then the ”user” tries to minimize the expected training error with the model.

Another way to explain it is that one can think of adversarial training as crafting the best adversarial example on a model at each epoch and then including those examples in the training set for that epoch. The technical reasoning for the oversimplifications is explained in Appendix A.

The adversarial example constructed should intuitively be the best that it can be. Therefore, the adversarial example used in adversarial training is a PGD attack [109], as previous works have empirically shown that preventing against PGD attacks also leads to mitigation against other types of attacks. An exception to this is Auto-Attack [33], a fairly recent work that is designed to mitigate adversarial training. A more rigorous explanation to why the adversarial example should be the best it can be is also explained in Appendix A.

The added step of solving the inner function (crafting the adversarial example), unfortunately drastically increases the amount of time and resources to train the model. In fact, not only does each epoch take longer to train (due to the creation of the adversarial example), but empirically, training the model itself takes more epochs than if a regular training algorithm was used.

Indeed, this enormous jump in resources remains a major downside of adversarial training. This is also not only just an empirical observation but grounded in theory as well [147]. To combat this, we look toward fine-tuning. Fine-tuning is a widely adopted technique in the machine learning training toolbox [207, 24, 50, 56, 27]. In this methodology, instead of training a model from scratch, one starts with a model that is already trained on benign inputs. The model is then made robust using a similar methodology as regular adversarial training.

In this work, we follow through with this line of approach and propose a new training methodology that can also be used for fine-tuning. Our methodology utilizes a balance of two different types to achieve better results, at a cost of slightly increased training time. Our experiments on the GTSRB and CIFAR-10 dataset show that our methodology is able to

achieve better robustness all without compromising accuracy on unaltered inputs. On top of this, our proposed training procedure takes up comparable time as the previous fine-tuning approachs and significantly less than regular adversarial training.

Our contributions are as follows:

- We propose a training algorithm for adversarial finetuning. We show that our methodology improves benign accuracy and robustness without incurring much overhead in terms of training
- We perform an in-depth analysis of the different methods of training so future works can be better informed of their options
- We show through experimentation that our method provides better robustness than the previous method against the PGD attack and the CW attack while maintaining a high-level of benign accuracy. At the same time, our method still take a comparable amount of time to run.

6.2 Methodology

6.2.1 Vanilla Adversarial Fine Tuning

Because the work by Jeddi *et al.* [73] is the closest to our work and the state of the art, we describe their proposed technique in more detail. Similar to our work, the authors aim to utilize fine-tuning to make a model robust against adversarial examples. This involves taking a pre-trained model and then running adversarial training for a few epochs, with the resulting model ideally having similar accuracy as that resulting from regular adversarial training. The challenge with fine-tuning a model on examples different from the training example (e.g. adversarial examples) is that the model can forget the previous inputs and

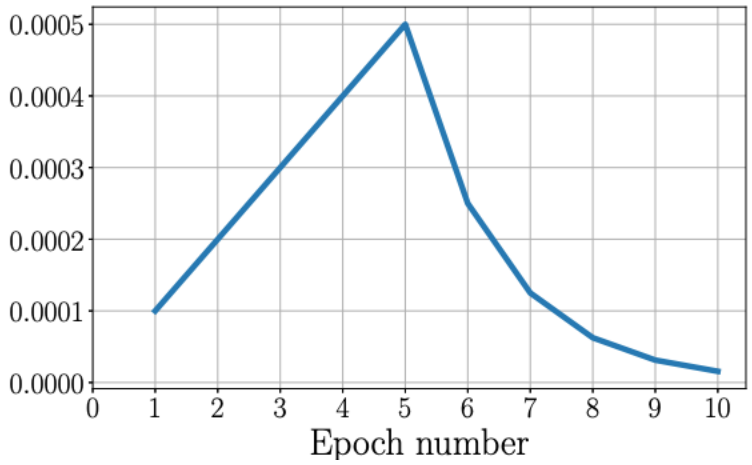


Figure 6.1: A sample training regimen provided by Jeddi *et al.* [73].

lead to catastrophic failure of the model. The authors propose to solve this through using a ”slow start, fast decay” method to help the model ”learn” adversarial examples without forgetting the benign inputs. This involves gradually increasing the learning rate for several epochs and then rapidly decaying the learning rate. An example they propose is to increase the learning rate by one step for 5 stages (e.g. 0.001, 0.002, 0.003, 0.004, 0.005) and then exponentially decaying it for 5 stages (Figure 6.1). Note that the training algorithm does not allow for much flexibility as the learning rate schedule is quite strict to carefully balance robust learning but also prevent catastrophic forgetting.

6.2.2 Our Methodology

In this section, we describe our proposed training methodology. As with vanilla adversarial fine-tuning, the process begins with training a clean model or obtaining a model that has already been trained. As one might expect, a model with a better starting accuracy is necessary for creating an end model with better performance.

In the first stage of training (Stage 1), we go through normal epochs of adversarial training, except that we first freeze all layers except for the head. We find that it is not necessary to randomly initialize the head and find that it leads to minor changes in final accuracy. After a certain point, we then unfreeze all layers and continue adversarial training on the full unfrozen model (Stage 2).

The amount of epochs to spend in each stage is a hyperparameter that is dependent on the model and the training dataset. We find that freezing all the layers except the head (Stage 1) helps maintain clean accuracy but the model gains robustness more slowly. Our experimentation also suggests that there is likely an upper bound, which makes sense since we have effectively limited the expressive power of the model. Stage 2 helps the model gain robustness much more rapidly but at a cost to benign accuracy. We find that the first epoch from moving to Stage 1 to Stage 2 causes the biggest drop in clean accuracy, but also the greatest gain in robustness. A graph showing the accuracies during a sample training run illustrates this point (Figure 6.2a)

We also find through experimentation that spending more time in Stage 1 before moving onto Stage 2 helps mitigate the drop in benign performance. As an example, we run our methodology on the GTSRB dataset for a total of 10 epochs. However, we vary the proportion of time spent in Stage 1, from 0% (0 epochs) to 100% (10 epochs). The resulting benign accuracies and robust accuracies of our models are shown in Figure 6.2b. The benign accuracy continues to increase, but it is important to note that the robust accuracy doesn't necessarily follow a similar pattern. We believe that two factors are working in the extremes. When we run Stage 2 only, the model experiences catastrophic failure, especially with such few epochs, and is unable to learn robustness well and also forget previously learned patterns. When we run Stage 1 only, the relevant factor is that the amount of epochs we run for are too small for the model to properly be trained against adversarial examples. To help illustrate this point, we run another set of training sessions in which we do

not spend any time on Stage 2 and only alter the amount spent on Stage 1. As shown, given enough epochs, the model can learn robustness without losing benign accuracy, suggesting a cause of the effect shown in Figure 6.2b.

For most models, we believe the ideal training procedure would be between the two extremes.

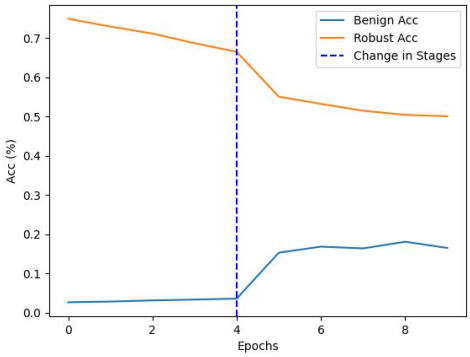
Remark 1. *Our procedure provides flexibility through balancing time spent in the two stages. By tuning the hyperparameter of training in the two stages, a user can decide between total time and computation cost, desired benign accuracy, and desired robustness on a model by model level.*

6.3 Evaluation

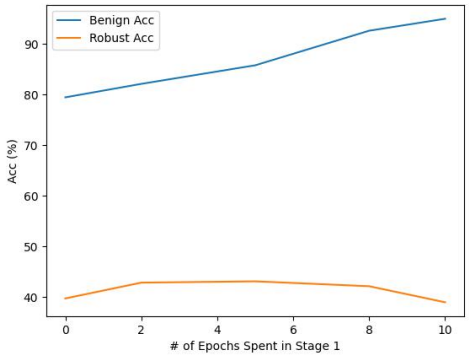
6.3.1 Evaluation Metrics

We compare our proposed framework against vanilla adversarial fine-tuning. For fair comparison, we utilize two metrics that are used in the original paper, which also happen to be standard metrics in the community. The first is accuracy on the original benign data which we will call benign accuracy. The second is accuracy under a PGD based attack, which we call robustness. A higher robustness value is ideal - a robustness value of 100 means that every attack is unsuccessful. We run attack settings under three PGD based attack settings. $\epsilon = 8/255$ and $\epsilon = 10/255$ are standard comparison metrics. We also include a third metric, a PGD attack with $\epsilon = 16/255$ to show an extreme attack scenario. Because AutoAttack [33] is designed to mitigate all adversarial training scenarios, we do not include it in our evaluation.

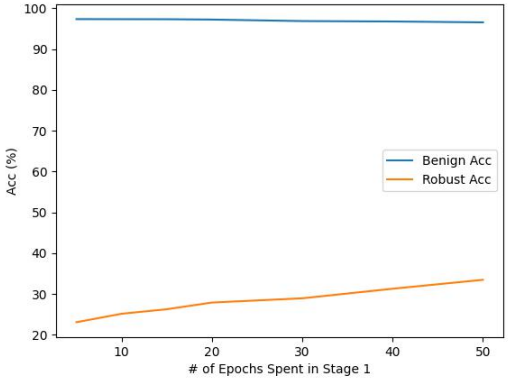
We run our experiments on two different datasets: GTSRB and CIFAR-10. All results are averaged over 3 different trials.



(a) Change from Stage 1 to Stage 2



(b) Percentage of epochs (GTSRB)



(c) GTSRB Stage 1 only

Figure 6.2: Graphs showing the results of various training experiments. Figure 6.2a shows the benign accuracy and robustness accuracy during a sample training session on GTSRB. The blue dashed line represents the change from Stage 1 and Stage 2. Notice the drop in benign accuracy. Graphs showing the results of various training experiments. Figure 6.2b shows the effect of benign accuracy and robustness when we vary the percentage of training epochs spent in Stage 1. Figure 6.2c shows the effect of benign accuracy and robustness when we train a model on GTSRB using just Stage 1.

To start, we trained a regular model from scratch. We utilized a VGG model architecture (VGG-16) for the GTSRB dataset and a ResNet architecture (Resnet-34) for CIFAR-10. We achieve a 97.11% regular accuracy on GTSRB and a 94.84 % accuracy on the CIFAR dataset. As a point of comparison, the base model for GTSRB obtains a 15.47 % accuracy

	GTSRB			
	Vanilla FT	Our Method (10 epochs)	Our Method (25 Epochs)	Regular Model
Benign Accuracy	81.03 (SD 0.02)	85.83 (SD 0.04)	84.29(SD 0.02)	97.11
PGD (8/255)	31.57 (SD 0.04)	43.1 (SD 0.02)	51.27 (SD 0.04)	15.47
PGD (10/255)	23.37 (SD 0.03)	33.53 (SD 0.01)	41.54 (SD 0.01)	6.94
PGD (12/255)	6.65 (SD 0.02)	13.53 (SD 0.01)	19.65 (SD 0.02)	0.47

	CIFAR-10		
	Vanilla FT	Our Method	Regular Model
Benign Accuracy	92.35 (SD 0.02	93.71 (SD 0.28)	94.84
PGD (8/255)	61.26 (SD 0.07)	79.08 (SD 0.27)	19.62
PGD (10/255)	51.19 (SD 0.04)	73.55 (SD 0.31)	12.0
PGD (12/255)	26.42 (SD 0.02)	54.32 (SD 0.27)	2.0

Table 6.1: Accuracies for our trained models on the PGD attack. In terms of both benign accuracy and accuracy under attack, our models match or exceed that created by vanilla adversarial fine-tuning

when faced with an adversarial attack under PGD with $\epsilon = 8/255$, while the base model for CIFAR obtains an accuracy of 19.62%. For all adversarial training methodologies, we utilize a PGD attack with $\epsilon = 8/255$ and allow for 20 iterations.

6.3.2 Accuracy

It is not often to see a tradeoff between benign accuracy and robustness - that is to increase a model's robustness, one has to often sacrifice accuracy on normal data. However, we find with our methodology, we are able to achieve better results in both benign accuracy and robustness across both datasets. As mentioned in Section 6.2, the amount of time spent in Stage 1 during training affects the final results.

As a fair comparison point, we show results after running for 10 epochs (5 in Stage

1 and 5 in Stage 2), which is the same number of epochs run for the vanilla adversarial fine-tuning. Our results for the GTSRB dataset is shown in Table 6.1 Using the above training procedure, we obtain a benign accuracy of an average of 85.83% with a robustness accuracy of 43.1%. Both these numbers exceed the accuracies obtained with the original vanilla fine-tuning procedure. To show the potential for our methodology, we also showcase results after running for 25 epochs to show what could be possible. For instance, running for 20 epochs in Stage 1 and 5 in Stage 2 drastically increases the robustness of the model. It is worth keeping in mind that both methodologies are well under the time it takes to run full adversarial training (Section 6.3.3)

To show that our model isn't somehow "overfitting" on PGD-based adversarial attacks, we also report accuracies while launching the Carlini-Wagner (CW) attack [16] on the models. For these attacks, the tradeoff between weighting amount of distortion (L_2 norm) versus strength of the adversarial attack is a hyperparameter that needs to be meticulously tuned. Stronger models will require adversarial attacks to have more distortion to reach the same level of success. For our evaluation, we try to get as close to 0% as possible while not reaching it and trying to keep all the models at roughly the same accuracy for a direct comparison. Table 6.2 shows our results, which shows that our model is able to resist the Carlini-Wagner attack as well.

Remark 2. *Our methodology matches or surpasses the original adversarial fine-tuning methodology in terms of robustness and benign accuracies.*

6.3.3 Timing

Because a major advantage and the entire purpose of adversarial fine-tuning is the time saved in training, we also evaluate on the timing. We ensure that the same amount of GPU

	Accuracies under CW Attack		
	Vanilla FT	Our Method	Regular Model
L_2 norm (GTSRB)	0.0078	0.013	0.0063
Robust Accuracy % (GTSRB)	4.02	4.41	4.51
L_2 norm (CIFAR10)	0.04	0.04	0.02
Robust Accuracy % (CIFAR10)	0.14	0.52	0.83

Table 6.2: The results of the Carlini-Wagner attack on our models versus compared to vanilla fine-tuning. We are able to match or exceed performance on both GTSRB and CIFAR10

	Time to train (seconds)
Vanilla FT (GTSRB)	640
Our method (GTSRB)	515
Regular AT (GTSRB)	20000
Vanilla FT (CIFAR-10)	10100
Our Method (CIFAR-10)	12000
Regular AT (CIFAR-10)	42000

Table 6.3: Average time to train each model averaged over 3 runs. Notice the sharp discrepancy between fine-tuning (FT) and regular adversarial training (AT)

cores are used throughout each dataset to allow for a fair comparison. One GPU core was used for GTSRB training while two were used for CIFAR-10. Once again, we run our results 3 times and average the results.

Depending on the exact training regimen used, the exact time differs. For instance, our model on GTRSB in which we utilize the same number of total epochs runs faster than vanilla adversarial fine tuning. However, our model on CIFAR-10 which we trained for a total of 22 epoch, takes a bit longer - 12000 seconds versus 10100 seconds. However, the key is that our approach is still orders of magnitude faster than full adversarial training, which takes roughly 42000 seconds on CIFAR-10 and roughly 20000 seconds on GTSRB.

Remark 3. *In terms of timing, our methodology is close to the original fine-tuning methodology. More importantly, it is still much faster than regular adversarial training, and maintains the "spirit" of fine-tuning.*

6.3.4 Limitations

We note a few limitations for our defenses. For one, we were unable to run on larger datasets like ImageNet. We also did not evaluate on black box attacks. Nevertheless, a few points remains: one, against the attacks evaluated in the original paper, our fine-tuning approach attains a better result. two - as a sanity check, we performed adapative attacks and even ran on extreme attack scenarios (i.e. PGD with a high ϵ value and CW with a high level of distortion) to help validate our results.

In short, though our work shows strong robustness against the attacks mentioned in this paper, it is not clear how well it works on other attacks. This is an interesting follow-up work to consider.

6.4 Conclusion

In this work, we propose a new training methodology for fine-tuning a model against adversarial examples. Our methodology involves balancing between two stage, which offers flexibility based on model type, desired accuracy, and desired computation cost. Our evaluation on the GTSRB and CIFAR-10 datasets show that our techniques exceed previous adversarial fine-tuning work, all while maintaining comparable training time.

CHAPTER 7

Conclusion

In this dissertation, we explored the different ways that adversaries can target machine learning and deep learning models. We showed that across different types of attacks, such as inference based and training based attacks, there are still missing gaps in terms of knowledge and defenses. In every scenario, we aimed to fill the gap in the knowledge and helped contribute to meaningful defenses against the various types of attacks. We can now revisit the questions posed in Chapter 1.

RQ 1: What are some current threats to models during inference time and can we build effective defenses? At the time this work was done, we realized there was a gap in understanding the effectiveness of adversarial examples on sensor fusion models. In Chapter 3, we showed that it is possible to conduct adversarial attacks on sensor fusion models with only modifying one of the inputs, namely the image input. We were able to construct a variety of attacks and tested some defenses, putting forth recommendations such as adversarial training.

In Chapter 6, we put forth a new training methodology for adversarial training that utilizes a pre-trained model and finetuning it to provide robustness against adversarial examples. Our evaluations on CIFAR-10 and GTSRB showed that it surpasses state of the art fine-tuning methods in terms of accuracy and matches accuracies provided by regular

adversarial training. Equally as importantly, our technique also takes much less time than regular adversarial training.

RQ 2: How can we improve defenses against attacks during training phase? Our work showed that there was a gap in the study of backdoor attacks, namely in the frequency domain of images. We showed in Chapter 4 that taking advantage of the frequency domains allows us to effectively detect backdoor triggers. We also show a technique to craft stronger attacks that do not leave high-frequency residues, but also show that taking these attacks into account allows us to build stronger defenses.

RQ 3: How can we provide a defense against anomalous inputs in industry systems? We showed in Chapter 5 that by utilizing an autoencoder based architecture, we were able to construct an anomaly detector that fit all of Ericsson's needs and metrics and was able to maintain competitive accuracies to traditional methods (DBSCAN and MAD). Not only that, but our methodology saves memory and time compared to the previous methods.

7.1 Limitations and Future Work

No work is perfect, and this dissertation is included in that statement. A major limitation to this work (as many) is that we could not run our experiments on as many cases as we had wanted. For example, in Chapter 3, we were limited by the availability of sensor fusion models at the time and lack of authors' responses. Throughout the work, we also were constrained by the fact that we could not test on every available dataset.

Nevertheless, security is a rapidly evolving field - with attacks and defenses constantly playing a cat and mouse game to "beat" the other side. The security of machine learning is no different and we believe that we have contributed to move the aforementioned cat-

and-mouse game forward. In fact, in some cases, our work has been left obsolete through the passage of time, but we still believe we have made an important contribution. Since our work shown in Chapter 3, several works have explored the robustness of sensor fusion models [172, 22]. In another example, our frequency work shown in Chapter 4 has inspired new generation of attacks [203, 195] that can surpass even the one posited in our work. Other works that have drawn upon ours include attacks in new areas [59, 107] and newly inspired defenses [106, 176].

We also remark, as stated in Chapter 1, the work presented in the dissertation is not the only security concerns faced by machine learning systems. Even during the time that this work was performed and written, new attacks have emerged, especially in the privacy space. For example, Carlini *et al*. [15] found that large language neural networks (like used by Google in their search recommendations) can memorize training inputs, leading to privacy concerns. Other privacy works include those that found one can regenerate training examples [201] and other membership inference attacks [152, 13, 123, 63, 165, 72, 2, 143, 188, 193].

7.2 Concluding Remarks

It is an indelible fact that machine learning systems are here to stay, and almost certainly become more and more ubiquitous. As the attack surface area continues to grow, it becomes difficult to assure the expected security of said systems.

This dissertation proposes practical solutions to understand some potential lapses in some machine learning systems and provides some potential suggestions for defenses. We hope that this work is just one stepping stone in the continued study for more secure and robust machine learning systems.

APPENDIX A

Technical Formality on Optimizing the Adversarial Training Equation

In Chapter 6, we define adversarial training as solving the following minimax equation:

$$\min_{\theta} \frac{1}{|S|} \sum_{(x,y)\in S} \max_{\delta\in\Delta(x)} L(F_\theta(x+\delta), y)$$

where L is the loss function with respect to a classifier F with model parameters θ taking in input x and output y. The adversary tries to add a perturbation δ to x bounded by some bound $\Delta(x)$ and S is the training set that we take the expected loss over.

In the chapter, we oversimplified and went straight to the end result of how to solve this function in practice. Here is the more formal breakdown.

Similar to how one would train a model normally, the way to solve the above minimax problem in practice is to use a gradient descent algorithm (like stochastic gradient descent with minibatch B and learning rate α) to update the model parameters θ:

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{(x,y)\in B} \nabla_\theta \max_{\delta\in\Delta(x)} L(F_\theta(x+\delta), y)$$

The gradient of the inner maximization function can be taken by applying Danskin's Theorem, which states (for the purpose of this) that finding the gradient of the maximum

function is given by the gradient of said function evaluated at the maximum.

Thus, we can define δ^* as follows:

$$\delta^* = \arg\max_{\delta \in \Delta(x)} L(F_\theta(x+\delta), y)$$

This is the exact procedure for finding the best adversarial example(!) This also allows us to solve the inner maximization problem without a dependence on θ:

$$\nabla_\theta \max_{\delta \in \Delta(x)} L(F_\theta(x+\delta), y) = \nabla_\theta L(F_\theta(x+\delta^*), y)$$

.

We now see why the simplified procedure shown in Chapter 6 where we alternate between finding an adversarial example and then training over said examples works.

An important note is worth mentioning: because we are using gradient descent to find δ^*, we are not finding the *true* maximum and Danskin's Theorem technically only applies in theory. Nevertheless, empirical studies shows that a local maximum is usually good enough. However, the strength of the model *does* depend on how close δ^* is to the maximum and so in adversarial training, careful care should be given to try to maximize said adversarial example. In short, δ^* should be optimal enough such that an adversary cannot "do better" in terms of the maximization function after the model is trained.

BIBLIOGRAPHY

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016.

[2] Saeed Ranjbar Alvar, Lanjun Wang, Jian Pei, and Yong Zhang. Membership privacy protection for image translation models via adversarial knowledge distillation, 2022.

[3] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples, 2017.

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.

[7] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, page 16–25, New York, NY, USA, 2006. Association for Computing Machinery.

[8] Battista Biggio, Luca Didaci, Giorgio Fumera, and Fabio Roli. Poisoning attacks to compromise face templates. In *2013 International Conference on Biometrics (ICB)*, pages 1–7, 2013.

[9] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci. Poisoning adaptive biometric systems. In Georgy Gimel'farb, Edwin Hancock, Atsushi Imiya, Arjan Kuijper, Mineichi Kudo, Shinichiro Omachi, Terry Windeatt, and Keiji Yamada,

editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 417–425, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[10] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff, 2020.

[11] Geoffrey J Burton and Ian R Moorhead. Color and spatial structure in natural scenes. *Applied optics*, 26(1):157–170, 1987.

[12] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Zhuoqing Morley Mao. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. In *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS'19)*, London, UK, November 2019.

[13] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles, 2021.

[14] Nicholas Carlini and Hany Farid. Evading deepfake-image detectors with white- and black-box attacks, 2020.

[15] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Santa Clara, CA, August 2019. USENIX Association.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017.

[17] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text, 2018.

[18] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in neural information processing systems*, 32, 2019.

[19] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. Domain generalization by mutual-information regularization with pre-trained models, 2022.

[20] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[21] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4658–4664. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[22] JXingyu Chen, Zhengxiong Li, Baicheng Chen, Yi Zhu, Chris Xiaoxuan Lu, Zhengyu Peng, Feng Lin, Wenyao Xu, Kui Ren, and Chunming Qiao. Metawave: Attacking mmwave sensing with meta-material-enhanced tags. In *The 30th Network and Distributed System Security (NDSS) Symposium 2023*, 2023.

[23] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 699–708, 2020.

[24] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[25] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models, 2020.

[26] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, 2017.

[27] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[28] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017.

[29] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *Deep Learning and Security Workshop*, 2020.

[30] Dan Ciresan, Alessandro Giusti, Luca Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[31] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[32] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066. PMLR, 2019.

[33] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[34] Francesco Croce and Matthias Hein. Adversarial robustness against multiple and single l_p-threat models via quick fine-tuning of robust classifiers, 2021.

[35] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, Jeremy Dawson, and Nasser Nasrabadi. Smoothfool: An efficient framework for computing smooth adversarial perturbations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2665–2674, 2020.

[36] J. Dai, C. Chen, and Y. Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.

[37] Shaohua Ding, Yulong Tian, Fengyuan Xu, Qun Li, and Sheng Zhong. Trojan attack on deep generative models in autonomous driving. In Songqing Chen, Kim-Kwang Raymond Choo, Xinwen Fu, Wenjing Lou, and Aziz Mohaisen, editors, *Security and Privacy in Communication Networks*, pages 299–318, Cham, 2019. Springer International Publishing.

[38] H. Drucker, Donghui Wu, and V.N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[39] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116*, 2019.

[40] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Physical adversarial examples for object detectors. In *Proceedings of the 12th USENIX Conference on Offensive Technologies*, WOOT'18, page 1, USA, 2018. USENIX Association.

[41] Samuel G. Finlayson, John D. Bowers, Joichi Ito, Jonathan L. Zittrain, Andrew L. Beam, and Isaac S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[42] Vlad Firoiu, William F Whitney, and Joshua B Tenenbaum. Beating the world's best at super smash bros. with deep reinforcement learning. *arXiv preprint arXiv:1702.06230*, 2017.

[43] Forbes. Machine learning (ml) market size, share &; covid-19 impact analysis, by component (solution, and services), by enterprise size (smes, and large enterprises), by deployment (cloud and on-premise), by end-user (healthcare, retail, it and telecommunication, bfsi, automotive and transportation, advertising and media, manufacturing, and others), and regional forecast, 2022-2029. *Fortune Business Insights Market Research Report*, 2022.

[44] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International Conference on Machine Learning*, pages 3247–3258. PMLR, 2020.

[45] Song Fu, Shisheng Zhong, Lin Lin, and Minghang Zhao. A re-optimized deep auto-encoder for gas turbine unsupervised anomaly detection. *Engineering Applications of Artificial Intelligence*, 101:104199, 2021.

[46] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 113–125, New York, NY, USA, 2019. Association for Computing Machinery.

[47] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[48] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[49] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[50] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[51] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification, 2016.

[52] Matt Growkoot. This ai image fooled judges and won a photography contest.

[53] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[54] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems, 2019.

[55] Yong Guo, David Stutz, and Bernt Schiele. Improving corruption and adversarial robustness by enhancing weak subnets. *arXiv preprint arXiv:2201.12765*, 2022.

[56] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[58] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong, 2017.

[59] Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. Cater: Intellectual property protection on text generation apis via conditional watermarks, 2022.

[60] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[61] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

[62] Pengyue Hou, Ming Zhou, Jie Han, Petr Musilek, and Xingyu Li. Adversarial finetune with dynamically regulated adversary, 2022.

[63] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv.*, 54(11s), sep 2022.

[64] Lifeng Huang, Chengying Gao, Yuyin Zhou, Changqing Zou, Cihang Xie, Alan Yuille, and Ning Liu. Upc: Learning universal physical camouflage attacks on object detectors, 2019.

[65] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations, 2019.

[66] Yi Huang, Adams Wai Kin Kong, and Kwok-Yan Lam. Adversarial signboard against object detector. In *British Machine Vision Conference*, 2019.

[67] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 387–395, 07 2018.

[68] IARPA. Trojans in artificial intelligence (trojai), Feb 2019.

[69] IBM. Ibm global ai adoption index 2022, 2022.

[70] iTrust Labs. Datasets, 2020.

[71] Gauri Jagatap, Ameya Joshi, Animesh Basak Chowdhury, Siddharth Garg, and Chinmay Hegde. Adversarially robust learning via entropic regularization. *Frontiers in artificial intelligence*, 4:780843, 2022.

[72] Ismat Jarin and Birhanu Eshete. Miashield: Defending membership inference attacks via preemptive exclusion of members, 2022.

[73] Ahmadreza Jeddi, Mohammad Javad Shafiee, and Alexander Wong. A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning. *CoRR*, abs/2012.13628, 2020.

[74] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks, 2020.

[75] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks, 2021.

[76] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning, 2021.

[77] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 267–280, New York, NY, USA, 1988. Association for Computing Machinery.

[78] Taewan Kim and Joydeep Ghosh. On single source robustness in deep fusion models. In *NeurIPS*, 2019.

[79] P. Kiourti, K. Wardega, S. Jha, and W. Li. Trojdrl: Evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[80] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer retraining is sufficient for robustness to spurious correlations, 2022.

[81] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

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

[83] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, abs/1712.02294, 2017.

[84] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

[85] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[86] Tencent Keen Security Lab. Experimental security research of tesla autopilot. Technical report, Tencent Keen Security Lab, 2019.

[87] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. *SIGCOMM Comput. Commun. Rev.*, 34(4):219–230, aug 2004.

[88] M.R. Leadbetter. On a basis for 'peaks over threshold' modeling. *Statistics & Probability Letters*, 12(4):357–362, 1991.

[89] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[90] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defense against general poisoning attacks, 2020.

[91] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies, 2015.

[92] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. *Advances in neural information processing systems*, 32, 2019.

[93] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, 2019.

[94] Shaofeng Li, Benjamin Zi Hao Zhao, Jiahao Yu, Minhui Xue, Dali Kaafar, and Haojin Zhu. Invisible backdoor attacks against deep neural networks. *arXiv preprint arXiv:1909.02742*, 2019.

[95] Shun Li and Jin Wen. A model-based fault detection and diagnostic methodology based on pca method and wavelet transform. *Energy and Buildings*, 68:63–71, 01 2014.

[96] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data, 2021.

[97] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[98] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust high-dimensional linear regression, 2016.

[99] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *AISec*, AISec '17, page 91–102, New York, NY, USA, 2017. Association for Computing Machinery.

[100] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. Opprentice: Towards practical and automatic anomaly detection through machine learning. *Proceedings of the 2015 Internet Measurement Conference*, 2015.

[101] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1), mar 2012.

[102] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, pages 273–294. Springer, 2018.

[103] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1265–1282, New York, NY, USA, 2019. Association for Computing Machinery.

[104] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2018.

[105] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017.

[106] Zeyan Liu, Fengjun Li, Zhu Li, and Bo Luo. Loneneuron: A highly-effective feature-domain neural trojan using invisible and polymorphic watermarks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2129–2143, New York, NY, USA, 2022. Association for Computing Machinery.

[107] Chengxiao Luo, Yiming Li, Yong Jiang, and Shu-Tao Xia. Untargeted backdoor attack against object detection, 2022.

[108] Shiqing Ma, Yingqi Liu, G. Tao, W. Lee, and X. Zhang. Nic: Detecting adversarial samples with neural network invariant checking. In *NDSS*, 2019.

[109] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[110] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, mar 2002.

[111] Shike Mei and Xiaojin Zhu. The Security of Latent Dirichlet Allocation. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 681–689, San Diego, California, USA, 09–12 May 2015. PMLR.

[112] Byeongjun Min, Jihoon Yoo, Sangsoo Kim, Dongil Shin, and Dongkyoo Shin. Network anomaly detection using memory-augmented deep autoencoder. *IEEE Access*, 9:104695–104706, 2021.

[113] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3578–3586. PMLR, 10–15 Jul 2018.

[114] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[115] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[116] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and P. Frossard. Universal adversarial perturbations. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017.

[117] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[118] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[119] Konda Reddy Mopuri, Aditya Ganeshan, and R. Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41, 01 2018.

[120] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks, 2018.

[121] Nour Moustafa and Jill 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)*, pages 1–6, 2015.

[122] Bingxu Mu, Zhenxing Niu, Le Wang, Xue Wang, Rong Jin, and Gang Hua. Progressive backdoor erasing via connecting backdoor and adversarial attacks, 2022.

[123] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization, 2018.

[124] Blaine Nelson and Anthony D. Joseph. Bounding an attack ' s complexity for a simple learning model. In *First Workshop on Tackling Computer Systems Problems with MachineLearning Techniques (SysML)*, 2006.

[125] Andrew Ng. Ai is the new electricity, Jan 2017. Remarks by Dr. Andrew Ng at Stanford MSx Future Forum.

[126] H.D. Nguyen, K.P. Tran, S. Thomassey, and M. Hamad. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, 2021.

[127] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[128] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 85–99, 2020.

[129] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, Xiapu Luo, and Ting Wang. TrojanZoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *2022 IEEE 7th European Symposium on Security and Privacy*. IEEE, jun 2022.

[130] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015.

[131] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C. Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection, 2018.

[132] R. Perdisci, D. Dagon, Wenke Lee, P. Fogla, and M. Sharif. Misleading worm signature generators using deliberate noise injection. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15 pp.–31, 2006.

[133] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Deep k-nn defense against clean-label data poisoning attacks, 2020.

[134] Anastaslia Poiner, David Wright, Gina Schaefer, Kartik Thopalli, Tanya Telford, and Tanya Urbanik. Automation with intelligence, 2022.

[135] PyTorch. Bootstrapping a multimodal project using mmf, a pytorch powered multimodal framework, 2020.

[136] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[137] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy, 2020.

[138] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial training can hurt generalization, 2019.

[139] Jee Rim. Introducing new scaled algorithms for improved outlier detection, 2017.

[140] Rui Zhang and Quanyan Zhu. A game-theoretic analysis of label flipping attacks on distributed support vector machines. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2017.

[141] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[142] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11957–11965, 2020.

[143] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. Sok: Let the privacy games begin! a unified treatment of data inference privacy in machine learning, 2022.

[144] Ahmed Salem, Yannick Sautter, Michael Backes, Mathias Humbert, and Yang Zhang. Baaan: Backdoor attacks against autoencoder and gan-based machine learning models, 2020.

[145] Esha Sarkar, Hadjer Benkraouda, and Michail Maniatakos. Facehack: Triggering backdoored facial recognition systems using facial characteristics. *arXiv preprint arXiv:2006.11623*, 2020.

[146] Dr. Eric Schmidt, Hon. Robert O. Work, Safra Catzand Dr. Steve Chien, Hon. Mignon Clyburn, Christopher Darby, Dr. Kenneth Ford, Dr. Jose Marie Griffiths, Dr. Eric Horvitz, Andrew Jassy, Gilman Louie, Dr. William Mark, Dr. Jason Matheny, Hon. Katharina McFarland, and Dr. Andrew Moore. Interim report november 2019. Technical report, National Security Commission On Artificial Intelligence, 2019.

[147] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. *Advances in neural information processing systems*, 31, 2018.

[148] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You autocomplete me: Poisoning vulnerabilities in neural code completion, 2020.

[149] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Explanation-guided backdoor poisoning attacks against malware classifiers, 2021.

[150] Zeyang Sha, Xinlei He, Pascal Berrang, Mathias Humbert, and Yang Zhang. Fine-tuning is all you need to mitigate backdoor attacks. *arXiv preprint arXiv:2212.09067*, 2022.

[151] Iman Sharafaldin, Arash Habibi Lashkari, and Ali Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 01 2018.

[152] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[153] Shoaib Ahmed Siddiqui and Thomas Breuel. Identifying layers susceptible to adversarial attacks, 2021.

[154] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[156] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[157] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[158] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[159] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery.

[160] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 877–894. USENIX Association, August 2020.

[161] Lichao Sun. Natural backdoor attack on text data, 2021.

[162] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[163] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[164] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[165] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by {Self-Distillation} through a novel ensemble architecture. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1433–1450, 2022.

[166] Markus Thill, Wolfgang Konen, Hao Wang, and Thomas Bäck. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, 112:107751, 2021.

[167] S. Thys, W. V. Ranst, and T. Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 49–55, 2019.

[168] DJ Tolhurst, Y_ Tadmor, and Tang Chao. Amplitude spectra of natural images. *Ophthalmic and Physiological Optics*, 12(2):229–232, 1992.

[169] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018.

[170] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.

[171] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy, 2018.

[172] James Tu, Huichen Li, Xinchen Yan, Mengye Ren, Yun Chen, Ming Liang, Eilyan Bitar, Ersin Yumer, and Raquel Urtasun. Exploring adversarial robustness of multisensor perception systems in self driving, 2021.

[173] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[174] Julian von Schleinitz, Michael Graf, Wolfgang Trutschnig, and Andreas Schröder. Vasp: An autoencoder-based approach for multivariate anomaly detection and robust time series prediction with application in motorsport. *Engineering Applications of Artificial Intelligence*, 104:104354, 2021.

[175] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, B. Viswanath, H. Zheng, and B. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy)*, pages 707–723, 2019.

[176] Haotao Wang, Junyuan Hong, Aston Zhang, Jiayu Zhou, and Zhangyang Wang. Trap and replace: Defending backdoor attacks by trapping them into an easy-to-replace subnetwork, 2022.

[177] Shaojie Wang, Tong Wu, and Yevgeniy Vorobeychik. Towards robust sensor fusion in visual perception, 2020.

[178] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems, 2021.

[179] Yuxin Wen, Jiehong Lin, Ke Chen, and Kui Jia. Geometry-aware generation of adversarial and cooperative point clouds. *CoRR*, abs/1912.11171, 2019.

[180] Emily Wenger, Josephine Passananti, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks on facial recognition in the physical world. *arXiv preprint arXiv:2006.14580*, 2020.

[181] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems*, 31, 2018.

[182] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

[183] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor, 2021.

[184] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *international conference on machine learning*, pages 1689–1698. PMLR, 2015.

[185] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[186] Cong Xu, Dan Li, and Min Yang. Adversarial momentum-contrastive pre-training. *Pattern Recognition Letters*, 160:172–179, 2022.

[187] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 187–196. International World Wide Web Conferences Steering Committee, 2018.

[188] Nuo Xu, Binghui Wang, Ran Ran, Wujie Wen, and Parv Venkitasubramaniam. NeuGuard: Lightweight neuron-guided defense against membership inference attacks. In *Proceedings of the 38th Annual Computer Security Applications Conference*. ACM, dec 2022.

[189] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019.

[190] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks, 2017.

[191] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in neural information processing systems*, 33:8588–8601, 2020.

[192] Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. Design of intentional backdoors in sequential models, 2019.

[193] Ziqi Yang, Lijin Wang, Da Yang, Jie Wan, Ziming Zhao, Ee-Chien Chang, Fan Zhang, and Kui Ren. Purifier: Defending data inference attacks via transforming confidence scores, 2022.

[194] Youngjoon Yu, Hong Joo Lee, Byeong Cheon Kim, Jung Uk Kim, and Yong Man Ro. Investigating vulnerability to adversarial examples on multimodal data fusion in deep learning, 2020.

[195] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. Narcissus: A practical clean-label backdoor attack with limited information. *arXiv preprint arXiv:2204.05255*, 2022.

[196] Yi Zeng, Han Qiu, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. *arXiv preprint arXiv:2012.07006*, 2020.

[197] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and N. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI*, 2019.

[198] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2017.

[199] H. Zhang and J. Wang. Towards adversarially robust object detection. In *ICCV*, pages 421–430, 2019.

[200] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[201] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[202] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1989–2004, New York, NY, USA, 2019. Association for Computing Machinery.

[203] Zhendong Zhao, Xiaojun Chen, Yuexin Xuan, Ye Dong, Dakui Wang, and Kaitai Liang. Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15213–15222, June 2022.

[204] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. One-class adversarial nets for fraud detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1286–1293, Jul. 2019.

[205] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4433–4439. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[206] Yingjie Zhou, Xucheng Song, Yanru Zhang, Fanxing Liu, Ce Zhu, and Lingqiao Liu. Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2021.

[207] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2019.

[208] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.