

MEASURING AND ENHANCING THE SECURITY OF MACHINE LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Florian Tramèr

August 2021

© 2021 by Florian Simon Tramer. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This dissertation is online at: <https://purl.stanford.edu/yz747qq9787>

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Percy Liang

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Gregory Valiant

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

The surprising failure modes of machine learning systems threaten their viability in security-critical settings. For example, machine learning models are easily fooled by adversarially chosen inputs, and have the propensity to leak the sensitive data of their users.

In this dissertation, we introduce new techniques to proactively measure and enhance the security of machine learning systems. We begin by formally analyzing the threat posed by adversarial examples to the integrity of machine learning models. We argue that the security implications of these attacks has been overstated for many applications, yet demonstrate one application where these attacks are indeed realistic—for evading online content moderation systems. We then show that existing defense techniques operate in fundamentally limited threat models, and therefore cannot hope to prevent realistic attacks.

We further introduce new techniques for protecting the privacy of users of machine learning systems—both at training and deployment time. For training, we show how feature engineering techniques can substantially improve differentially private learning algorithms. For deployment, we design a system that combines hardware protections and cryptography to privately outsource machine learning workloads to the cloud. In both cases, we protect a user’s sensitive data from other parties while achieving significantly better utility than in prior work.

We hope that our results will pave the way towards a more rigorous assessment of machine learning models’ vulnerability against evasion attacks, and motivate the deployment of efficient privacy-preserving learning systems.

to Mariël

Acknowledgments

I don't quite recall *why* I originally wanted to do a PhD. I had done some research before graduate school, and loved some aspects of it but resented others. Ultimately, I knew that I had really enjoyed my time in school so far, and it thus felt like a good idea to extend this a bit. Various sources of online advice for prospective graduate students will warn you that this is the worst possible reason to want to do a PhD, and that you'll likely struggle a lot as a consequence. Given how things turned out, I'd conclude that: (1) most "advice" you find online is bad; (2) none of this would have been possible without the tremendous help and motivation I've received along the way.

First and foremost, I want to thank my advisor Dan Boneh for his invaluable guidance throughout my PhD. From the first day, Dan encouraged me to pursue research in an emerging area that neither of us had much familiarity with, and gave me absolute freedom and support to develop my own research interests and collaborations. Dan's relentless optimism, passion for research, and genuine trust in my own abilities have shaped me into a more confident and independent researcher. I strive to show my own future students the same freedom, conviction, and kindness.

The fellow students and visiting researchers from the Applied Cryptography Group have been great colleagues and friends during these past five years. Despite working in a somewhat orthogonal field, I received amazing feedback and learned a great deal about research from all of them: Alex Ozdemir, Ben Fisch, Benedikt Bünz, Benton Case, David Wu, Dima Kogan, Giancarlo Pellegrino, Henry Corrigan-Gibbs, Kamil Kluczniak, Lucjan Hanzlik, Neil Perry, Riad Wahby, Saba Eskandarian, Sam Kim, Sergio Benitez and Yan Michalevsky.

Benedikt, Dima, Saba and I started our PhDs at the same time, and it is incredible to look back at how far we've come as researchers and friends. I feel that we all drove each other to become more ambitious and effective researchers. They've always been there for incredible advice and ideas, for office gossip over coffee, or to listen to me ranting on some random topic over dinner or a drink.

I also want to extend a special thanks to Henry, who has been a constant source of guidance and inspiration. Regardless of the topic (what to research or teach, and how; how to write and present; where to live and work) or the setting (office, café or ski slope), Henry always seems to have a very

principled opinion to share, and I don't think I've ever gone wrong by heeding it.

My research collaborations with Giancarlo have taught me a lot on how to shape research projects and advise students. I've had the chance to apply these lessons when advising projects with some amazing students: Blanca Villanueva, Edward Chou, Evani Radiya-Dixit and Gili Rusak. A project with Gilli and Giancarlo, as well as Pascal Dupré, forms the basis of Chapter 3 of this dissertation.

My co-instructors for CS355—David Wu, Dima Kogan, Henry Corrigan-Gibbs, Katy Woo, Saba Eskandarian and Sam Kim—have taught me (and our students) everything there is to know about modern cryptography. Our (overly) long debates about course contents and learning objectives have helped me become a more engaged and effective teacher.

I've been able to count on the technical and moral support from many of Stanford's faculty members over the years. Greg Valiant, Moses Charikar and Percy Liang sat on the committee for my dissertation defense and have been great sources of research inspiration at various times during my PhD. I thank Mykel Kochenderfer for enthusiastically agreeing to chair my defense committee. On multiple occasions, David Mazières, John Mitchell, Matei Zaharia, Omer Reingold and Zakir Durumeric have given me extremely helpful feedback and candid criticism on my research and talks.

The staff in Gates always made sure to take care of any issues that would threaten to distract me from my research (and there were many)! Megan Harris and Ruth Harris handled everything from research supplies, office arrangements, travel reimbursements—always with the utmost kindness. Jam Kiattinant and Angela Cao made sure that my funding was always in order, and Jay Subramanian was there at all times to give great advice about the PhD requirements and various visa issues I would encounter along the way. Their help has been invaluable.

My research collaborators outside Stanford have surprisingly become too numerous to list here. I owe a great deal of thanks for this to Nicolas Papernot, who visited Stanford shortly after I started my PhD, and became a close collaborator and friend. Nicolas introduced me to many people over the years who would later become collaborators. His determination and optimism in research are inspiring and I could always count on incredible advice from him.

In the later stage of my PhD, I developed a very fruitful collaboration with Nicholas Carlini. I've learned a great deal from Nicholas about conducting rigorous and fair attack-oriented research. His relentless confidence and determination remain an incredible motivator. The work in Chapter 5 of this dissertation was done jointly with Nicolas, Nicholas, Jens Behrmann and Jörn-Henrik Jacobsen.

Kenny Patterson hosted me for a refreshing (albeit incredibly hot) summer at ETH Zürich. As Kenny was just starting a lab, I think I spent more time interacting with him during those three months than with any other collaborator over the entire duration of my PhD. In large parts due to his advice and support, I will soon be joining him as a colleague!

In the last year of my PhD, passionate discussions with Ilya Mironov were instrumental in helping

me partially shift my research focus, and inspired the work in Chapter 6 of this dissertation.

Before graduate school, Serge Vaudenay, Ola Svensson and Jean-Pierre Hubaux introduced me to research in Computer Science, and helped me develop a broad foundation in cryptography, theory and privacy that drives my research today. The year in which Jean-Pierre hosted me at EPFL before beginning my PhD remains one of my most productive and rewarding to date, and his continued support and advice throughout my academic career have been amazing.

I could always count on Ari Juels to share incredibly creative and interesting research ideas. Our early work on model stealing attacks with Fan Zhang, Michael Reiter and Tom Ristenpart ultimately drove me down the road of machine learning security that this dissertation is all about. I was also lucky to continue collaborating with Ari throughout my PhD, and our work provided refreshing breaks away from the world of machine learning when I needed them.

I want to thank Aleksander Mądry, Carmela Troncoso, and Ludwig Schmidt for a lot of valuable counsel and insights about the academic job search in the past year.

I would have never managed to get through with this PhD if not for the incredible moral support from all my family and friends. I was lucky to make some amazing friends at Stanford that have made sure to pull me away from work as often as possible for campus parties, roadtrips, camping adventures, and holidays. A special thanks to Begum & Tanay, Faidra & Kostis, Marie & Yannaï, Olivia & Toby and Tal & Dima for the unforgettable adventures we've had together.

When traveling for work or heading back home on vacation, I could always count on my friends for an eventful combination of dinners, sleepovers, parties, weddings and newborns. A huge thanks to all of them: Adrien, Axel, Christophe, Christelle & François, Conti & Eamon, Danee, Imke & Peter, Izatti & Daniel, Karin & Erik, Kristof, Florence & Axel, Laura & Romain, Lucille & Joachim, Maryam & Salman, Marceline & Timothée, Melissa & Sacha, Naomi & Romain, Titus, Saviz and Yonathan. I reserve a special thought for my dear friend Adrien Gaudard—in loving memory.

Helen and Tom Gracon were guardian angels that offered housing, food, time and constant professional and life advice. As I write this, it has become painfully clear that settling in after an international move is a lot more challenging when they aren't around to help. Thanks for everything!

In addition to all my family members, who always made sure I was well fed around Christmas season, I want to warmly thank Oscar, Annemiek, Dorine and Elsa for welcoming me into their family, and for their genuine support and wonderful hospitality over the years.

I want to thank my parents, Claudia & Martin, and my brothers, Joël & Lucas, who have encouraged and supported me for as far as I can remember. My early education instilled the skills that have made this PhD possible—a love for learning and reading, a decent mastery of English, and above all, perseverance. I cannot thank my parents enough for everything they've done for me.

Finally of course, thank you Mariël, quite simply for being there every step of the way. This dissertation is dedicated to you.

Contents

Abstract	iv
Acknowledgments	vi
1 Introduction	1
1.1 Overview of Results	2
1.2 Machine Learning Background	9
I The Security Threat of Adversarial Examples	13
2 The Threat Model of Adversarial Examples	15
2.1 The Expectimax Game	16
2.2 Adversarial Examples As a Necessary Attack Vector	18
2.3 Choosing a Perturbation Set	19
3 A Security Application: Evading Perceptual Ad-blockers	21
3.1 Preliminaries and Background	24
3.1.1 The Online Advertising Ecosystem	24
3.1.2 Perceptual Ad-blocking	24
3.1.3 Threat Model and Adversaries	26
3.2 Designing Perceptual Ad-blockers	27
3.2.1 General Architecture	27
3.2.2 Approaches to Ad Detection	28
3.3 Training a Page-based Ad-blocker	30
3.3.1 Data Collection	30
3.3.2 Evaluation and Results	31
3.4 Evaluating the Robustness of Perceptual Ad-blocking	33
3.4.1 Evaluation Setup	34

3.4.2	Accuracy and Performance of ML classifiers.	36
3.5	Attacking Ad Classifiers With Adversarial Examples	36
3.5.1	Attack Model	37
3.5.2	Overview of Attack Techniques and Results	37
3.5.3	Algorithms for Adversarial Examples	39
3.5.4	Results	40
3.6	Attacks Beyond Misclassification	44
3.6.1	Attacks Against Ad-blocker Actions	44
3.6.2	Attacks Against Page Segmentation	46
3.6.3	Attacks Against Training	48
3.7	Discussion	48
3.7.1	A New Arms Race	48
3.7.2	Strategic Advantage of Adversaries and Lack of Defenses	49
3.7.3	Beyond the Web and Vision	49
3.8	Related Work	50
3.9	Conclusion	51
4	Limitations of Defenses: Multiple Perturbation Types	52
4.1	Theoretical Limits to Multi-perturbation Robustness	55
4.1.1	Adversarial Risk for Multiple Perturbation Models	55
4.1.2	A Binary Classification Task	56
4.1.3	Small ℓ_∞ and ℓ_1 Perturbations are Mutually Exclusive	56
4.1.4	Small ℓ_∞ and Spatial Perturbations are Nearly Mutually Exclusive	58
4.1.5	Affine Combinations of Perturbations	62
4.2	New Attacks and Adversarial Training Schemes	67
4.3	Experiments	69
4.3.1	Results on MNIST	71
4.3.2	Results on CIFAR-10	71
4.3.3	First-order Adversarial Training and Gradient Masking on MNIST	73
4.3.4	Affine Adversaries	74
4.4	Discussion and Open Problems	75
5	Limitations of Defenses: Excessive Invariance	77
5.1	Norm-bounded Sensitivity and Invariance Attacks	79
5.2	The Sensitivity and Invariance Tradeoff	81
5.3	Generating Invariance-based Adversarial Examples on MNIST	83
5.3.1	Generating Model-agnostic Invariance-based Adversarial Examples	84
5.3.2	Evaluation	86

5.3.3	Trading Perturbation-robustness for Invariance-robustness	89
5.3.4	Natural Images	91
5.4	The Overly-robust Features Model	92
5.4.1	Formal Model and Analysis	92
5.4.2	Experiments	94
5.5	Discussion	95
5.6	Conclusion	96
5.7	Complete Set of Invariance Adversarial Examples	98

II Privacy-Preserving Machine Learning 102

6 Differentially Private Learning With Better Features 107

6.1	Preliminaries	109
6.1.1	Scattering Networks	109
6.1.2	Differentially Private Stochastic Gradient Descent	111
6.1.3	Differentially Private ScatterNet Classifiers	112
6.2	Evaluating Private ScatterNet Classifiers	113
6.2.1	Experimental Setup	114
6.2.2	Model Architectures	116
6.2.3	Results	117
6.2.4	Analysis of Hyper-parameters	119
6.3	How Do Handcrafted Features Help?	120
6.3.1	Smaller Models Are Not Easier to Train Privately	121
6.3.2	Models With Handcrafted Features Converge Faster <i>Without Privacy</i>	121
6.4	Towards Better Private Deep Learning	122
6.4.1	Improving Privacy by Collecting More Data	124
6.4.2	Transfer Learning: Better Features from Public Data	125
6.5	Additional Experiments	127
6.5.1	On the Effect of Batch Sizes in DP-SGD	127
6.5.2	Comparing DP-SGD and Privacy Amplification by Iteration	129
6.5.3	DP-SGD With Poisson Sampling	131
6.6	Conclusion and Open Problems	131

7 Slalom: Faster Private Inference With Trusted Hardware 133

7.1	Background	134
7.1.1	Problem Setting	134
7.1.2	Trusted Execution Environments (TEEs), Intel SGX, and a Strong Baseline	135
7.1.3	Outsourcing Outsourced Neural Networks and Freivalds' Algorithm	137

7.2	Formal Security Definitions	138
7.3	Slalom	140
	7.3.1 Quantization	140
	7.3.2 Verifying Common Linear Operators	142
	7.3.3 Input Privacy	144
7.4	Empirical Evaluation	145
	7.4.1 Implementation	145
	7.4.2 Setup	146
	7.4.3 Neural Network Details	147
	7.4.4 Results	147
	7.4.5 Results on a Standard CPU	151
	7.4.6 Parallelization	151
7.5	Challenges for Verifiable and Private Training	152
7.6	Conclusion	153

III Conclusion **154**

List of Tables

3.1	Attack strategies on perceptual ad-blockers	33
3.2	Evaluation of ad-classifiers	34
3.3	Evaluation data for attacks on perceptual ad-blockers	35
4.1	Evaluation of MNIST models trained on Linf, L1, L2 and RT attacks	71
4.2	Evaluation of CIFAR-10 models trained on Linf, L1 and RT attacks	72
4.3	Breakdown of all attacks on MNIST models	72
4.4	Breakdown of all attacks on CIFAR-10 models	73
4.5	Evaluation of affine attacks	74
5.1	Success rate of invariance adversarial examples against humans	87
5.2	Agreement between models and humans on invariance adversarial examples	88
5.3	Model accuracy on sensitivity-based adversarial examples	89
5.4	Robust accuracy as a function of perturbation size during training.	90
6.1	Accuracy of differentially private models with handcrafted ScatterNet features	108
6.2	Effect of feature normalization on the accuracy of non-private ScatterNet models	113
6.3	Hyper-parameters for the evaluation of differentially private classifiers	115
6.4	Architecture of end-to-end CNNs	116
6.5	Architecture of CNN models fine-tuned on ScatterNet features	117
6.6	Test accuracy for models trained without privacy	117
6.7	Accuracy variability across hyper-parameters	119
6.8	Number of trainable parameters per model	121
6.9	Comparison of small and large CIFAR-10 CNNs	121
6.10	Comparison of DP-SGD with Poisson sampling and random shuffling.	131
7.1	Security guarantees and performance of different ML outsourcing schemes	137
7.2	Complexity of evaluating and verifying linear functions	143
7.3	Details of models used to evaluate Slalom	146

List of Figures

2.1	The expectimax security game for adversarial examples	16
3.1	Ad-blocker privilege hijacking	23
3.2	The AdChoices logo	24
3.3	The architecture of a perceptual ad-blocker	27
3.4	Perceptual ad-blocking elements	30
3.5	Activation maps of a page-based ad-blocker	32
3.6	Adversarial examples for element-based classifiers	40
3.7	Adversarial examples for frame-based classifiers	41
3.8	Attack on the Percival browser	42
3.9	Code snippet for universal evasion attack	43
3.10	Universal adversarial examples for page-based ad-blockers on BBC.com	44
3.11	Universal adversarial examples for page-based ad-blockers	45
3.12	CSS obfuscation on Facebook.com	46
3.13	Image sprites of the AdChoices logo	47
3.14	Adversarial audio waveform	50
4.1	Multi-robustness trade-off on MNIST and CIFAR-10	54
4.2	Performance of the Sparse L1 Descent Attack for different choices of descent directions	69
4.3	Gradient masking for an adversarially trained MNIST model	74
4.4	Illustration of affine attacks	75
5.1	Decision boundaries and invariance-based adversarial examples	78
5.2	Illustration of distance-oracle misalignment	81
5.3	An ℓ_p norm fails to measure semantic similarity in images	83
5.4	Process for generating ℓ_0 invariant adversarial examples	85
5.5	Invariance-based adversarial examples on MNIST	87
5.6	Higher noise-robustness leads to higher vulnerability to invariance attacks	91
5.7	Robust accuracy against an affine adversary	95

5.8	Model-human agreement on successful invariance adversarial examples	96
6.1	Privacy-accuracy tradeoffs for ScatterNet classifiers	118
6.2	Median and maximum test accuracy with one hyper-parameter fixed	120
6.3	Convergence rates of private and non-private models	123
6.4	Trade-off between model accuracy and the size of the training set	126
6.5	Privacy-utility tradeoffs for transfer learning on CIFAR-10	127
6.6	Noise scale as a function of the sample rate	128
6.7	Convergence rate of DP-SGD for different batch sizes	129
6.8	Comparison of noise scales for DP-SGD and Privacy Amplification by Iteration . . .	130
6.9	Privacy-accuracy tradeoffs for DP-SGD and Privacy Amplification by Iteration . . .	130
7.1	The Slalom algorithms for verifiable and private neural network inference	141
7.2	Micro benchmarks on Intel SGX.	147
7.3	Verifiable and private inference with Intel SGX	148
7.4	Secure outsourcing of ResNet models with Intel SGX	149
7.5	Micro benchmarks on an untrusted CPU	150
7.6	Inference with integrity and privacy on an untrusted CPU	151
7.7	Multi-threaded micro benchmarks on an untrusted CPU	152

Chapter 1

Introduction

Data-driven decision systems—powered by recent and ongoing advances in deep learning—are set to play a central role in areas as diverse as self-driving [18], computer security [13], healthcare [72, 157], messaging [38] or smart home assistants [209].

Each of these applications faces clear security risks. For example, violations of the *integrity* of machine learning systems represents a major concern when these systems are deployed in adversarial settings. While deep learning models can extract rich statistical patterns that match or exceed human performance on a number of perceptual tasks, these models are surprisingly brittle to manipulation. Imperceptible perturbations to a model’s inputs (during training [16, 44] or evaluation [17, 95, 246]) can cause a model to produce arbitrarily incorrect outputs.

By virtue of being inherently driven by *data* (the “new oil” [68]), machine learning systems also raise a number of concerns for the *privacy* of their users. Indeed, building machine learning models often requires collecting and aggregating large amounts of sensitive user information such as medical images, personal messages or driving itineraries. Even if users entrust this information to the party that provides the machine learning system, their data could still leak to other users of the same system [31, 32, 235].

It is thus imperative to develop techniques to proactively mitigate these security and privacy risks. Yet, as a necessary first step, we must develop frameworks and tools to *measure* the extent to which these threats apply to existing systems. This dissertation addresses both of these challenges, by proposing new approaches for measuring and enhancing the security of machine learning.

In the first part of this dissertation, we qualitatively and quantitatively assess the risk of *evasion attacks* on deployed machine learning models. We formally analyze the threat model of a widely popularized class of attacks called *adversarial examples* [246], minimally perturbed inputs that fool machine learning systems. We show that data-driven content blocking systems on the Web are uniquely predisposed to these attacks, and that existing defense techniques are fundamentally

inadequate for preventing realistic threats.

In the second part of this dissertation, we develop techniques to protect the privacy of machine learning users. We propose new learning paradigms that provably prevent leaking of users’ training data (using the guarantees of *differential privacy* [66]), while achieving significantly higher utility than in prior work. We further design a system, Slalom, that efficiently outsources machine learning workloads to a remote untrusted cloud, while preserving the privacy of the user’s requests.

Overall, the results in this dissertation paint two contrasting pictures. On the one hand, enhancing the integrity of machine learning systems remains by-and-large an unsolved problem. As a result, certain security-critical applications of machine learning (for example for content blocking on the Web) will likely remain out of reach for the foreseeable future. On the other hand, we show that it is possible to design learning systems with strong privacy guarantees, at only moderate costs in performance.

1.1 Overview of Results

Part I: The Security Threat of Adversarial Examples

In the first part of this dissertation, we study the security threat posed by adversarial examples [17, 95, 246], maliciously perturbed inputs that cause machine learning models to fail. While prior work had doubted the relevance of these attacks in concrete security-relevant settings [89, 185], we demonstrate a compelling application of adversarial examples for evading content-moderation systems on the Web. We then introduce and analyze two intrinsic limitations of current defenses against adversarial examples, which limit the usefulness of these defenses in practice.

On threat models. The term “adversarial example”—introduced in the seminal work of Szegedy et al. [246]—has at times been used to refer to any type of adversarially manipulated input of a machine learning model [94]. This generic characterization is hard to work with, however, as it defines adversarial examples in terms of their malicious usage rather than in terms of intrinsic properties of these examples. In this dissertation, we will work with a more pragmatic and widely-used definition of adversarial examples: given a classifier f and an input x (sampled from some underlying data distribution), an adversarial example for x is a perturbed input \hat{x} that is misclassified by f , and that is perceptually “close” to x . Building classifiers that are *robust* to small perturbations of their inputs is a major unsolved challenge in machine learning today (as we will see, formalizing the notion of perceptual closeness is a major challenge in itself).

The study of adversarial examples is often motivated by the application of machine learning in security-sensitive or safety-critical applications. For example, prior work has shown that small perturbations to street signs could cause a self-driving car to crash [73, 74]; imperceptible audio commands can trigger voice assistants [28, 29]; and small printed noise patterns can fool facial

recognition software [228].

In Chapter 2, we define a formal threat model for adversarial examples, which is often left implicit in the literature. This threat model is characterized by a *security game* between a challenger (who builds a model) and an adversary (who aims to attack the model). Our formalization borrows from the prior work of Gilmer et al. [89] and Goodfellow [93].

We then argue that for the above security-sensitive applications (i.e., self-driving, voice assistants or facial recognition), the threat model of adversarial examples is unnecessarily restrictive. Adversarial examples are typically but one (relatively complex) way for an adversary to breach a system’s security or safety properties, and are by no means a *necessary* attack vector. In particular, the implicit restriction that an adversary can only add “small” perturbations to inputs is rather artificial. For example, an attacker could show a (real) STOP sign in their rear window to halt a self-driving car that is tailing them; play a malicious TV ad with a perfectly audible command that triggers a voice assistant when the owner is inattentive; or wear a prosthetic face-mask to bypass facial recognition systems [223]. In all of these scenarios, the attacker may succeed by showing *arbitrary* out-of-distribution inputs to a machine learning system.

A (real) security application: evading perceptual ad-blockers. The work we present in Chapter 3 is thus motivated by the following question:

Is there a security-sensitive task where an adversary is constrained to apply small perturbations to inputs when attempting to evade a classifier?

What should such a task look like? First, there must be a human-in-the-loop: the attacker’s goal should be to evade a classifier while ensuring that any human observer is oblivious to the attack (if there is no human observer, it is not clear what prevents the attacker from using an arbitrary input). Second, there must exist some distribution over inputs that the attacker cannot control (otherwise, the attacker can just pick an arbitrary misclassified input, a so-called “test-set attack” [89]).

We show that the task of content moderation on the Web perfectly matches this threat model. In this task, the goal of a machine learning model is to automatically filter and block online content that may be undesirable for users, e.g., advertisements or offensive media. In turn, an attacker’s goal is to take an input intended to be shown to users, and minimally modify that input so that it bypasses detection—a perfect use-case for adversarial examples!

Our study in Chapter 3 focuses on the use of machine learning for online ad-blocking. The growing use of ad-blockers such as Adblock Plus and uBlock has sparked a fierce arms race with publishers and advertising networks. Departing from the classical and brittle approach of detecting ads based on metadata, Storey et al. [244] first proposed the concept of a *perceptual ad-blocker*, that uses computer vision to emulate the way in which humans visually detect ads. If effective, such an ad-blocker would mark the end of the arms race, with the ad-blockers claiming victory. Over the

past years, popular ad-blockers such as Adblock Plus [203, 272] and the Brave browser [261] have experimented with the incorporation of perceptual signals.

We demonstrate that all machine learning algorithms that have been considered for perceptual ad-blocking can be evaded using imperceptible adversarial examples. These algorithms range from classical computer-vision techniques such as perceptual hashing to deep neural networks. The vulnerability of ad classifiers is exacerbated by the fact that ad-blockers operate *client-side*. An attacker can thus get full access to the ad-blocker code, and to the parameters of its machine learning models.

Our attacks can be used by web publishers or advertising networks to evade an ad-blocker with arbitrary ad content, without affecting the end-users’ perception of the ads or of other web content. We show how to create perturbations that (1) can be encoded as valid HTML elements; (2) are robust to content changes outside of the adversary’s control (i.e., perturbed ads evade blocking on any page where they appear); and (3) scale to thousands of pages and ads.

We further show that perceptual ad-blocking creates new web vulnerabilities. Specifically, we demonstrate a *content hijacking* attack wherein one user of a social media platform (e.g., Facebook) posts malicious content that fools an ad-blocker into blocking *other users’ content* on the platform.

Our attacks show that perceptual techniques will not mark the end of the ad-blocking arms race, as long as machine learning models remain vulnerable to perceptually small adversarial examples.

Limitations of defenses. Building an attack-resistant model for perceptual ad-blocking (or for other content moderation tasks) would require building a model that resists adversarial examples. Yet, building such a defense is a remarkably challenging problem. And it is not for a lack of trying: a large number of heuristic defense approaches have been proposed and have successively been broken [6, 26, 27, 258].

A first challenge is to formalize what it means for a defense to be robust. Our definition of an adversarial example above states that a perturbed input must be “perceptually close” to the original input. Yet, for many data types of interest, such as natural images, we do not know how to formally characterize perceptual similarity between inputs. Instead, prior work has opted to approximate this definition by considering only specific explicitly-defined distance metrics. A common approach is to aim for robustness against perturbations from a well-defined small set (e.g., all perturbations within some small ℓ_p ball [95, 159]).

Prior work has shown successful techniques for training classifiers that are robust to small perturbations from such a fixed set [71, 159, 207, 271]. In particular, adversarial training [159, 246] produces models with strong (empirical) robustness guarantees. Moreover, *certified defenses* [53, 83, 146, 207, 271] even achieve *provable* (but empirically weaker) robustness guarantees.

Despite this tremendous progress, these defense techniques cannot currently improve the practical security of machine learning models deployed in adversarial settings—such as the perceptual ad-blockers that form the subject of Chapter 3. Indeed, in Chapter 4 and Chapter 5, we highlight fundamental limitations of current defenses against adversarial examples. In fact, we show that

existing defenses may achieve robustness to perturbations that is neither *sufficient* nor *necessary*.

In Chapter 4, we introduce the problem of training neural networks that are robust to *multiple types* of small perturbations. Prior work has focused on building models that are robust (either empirically or provably) to adversarial perturbations from a single fixed set, e.g., perturbations of small ℓ_∞ norm [159, 207, 271]. While these models do attain some robustness to perturbations from this set, they remain entirely vulnerable to other types of small perturbations. For example, adversarial training against perturbations of small ℓ_∞ norm yields no robustness to perturbations of small ℓ_1 norm [230], and actually increases a model’s vulnerability. This leads us to the central problem considered in Chapter 4:

How can we achieve adversarial robustness to different perturbation types simultaneously?

To gain intuition about this problem, we first study a simple and natural synthetic classification task. We show that for this task, some perturbation types (e.g., perturbations of small ℓ_∞ norm or of small ℓ_1 norm) are *mutually exclusive*. That is, increasing robustness to one type of perturbations necessarily implies decreasing robustness to the other. The existence of such a trade-off for this simple synthetic classification task may explain its prevalence in more complex statistical settings.

To complement our formal analysis, we introduce new adversarial training schemes for multiple sets of small perturbations. Our best-performing scheme augments a model’s training data with worst-case perturbations from the union of these sets. We experimentally show that models trained against multiple perturbation types fail to achieve robustness competitive with that of models trained on each perturbation type individually. In particular, for the task of classifying handwritten digits [145], we find that a model trained to be robust to perturbations of small ℓ_∞ , ℓ_1 or ℓ_2 norm achieves only 50% robust accuracy. Thus, even this simple task is far from solved in a robust sense.

In summary, despite recent successes in achieving robustness to single perturbation types, many obstacles remain towards scaling existing techniques to richer combinations of small perturbations.

In Chapter 5, we take a step back and ask another seemingly benign question with far-reaching implications for the design of defenses against adversarial examples:

How large of a perturbation set should our models be made robust to?

A model clearly cannot be robust to unbounded perturbations, as these can change one object (e.g., a cat) into an object from another class (e.g., a dog). Prior work has mainly side-stepped this issue, because most defenses only scale to perturbations that are far too small to be perceptible to humans. Such defenses thus remain overly *sensitive* to small input perturbations.

For simple tasks such as digit classification, current defenses do achieve robustness to larger perturbations (of a single type). However, we show that by “over-optimizing” their robustness to perturbations, some defenses become excessively *invariant* to real semantics of the underlying task.

In Chapter 5 we expose a fundamental tradeoff between a model’s sensitivity and invariance. We find that by decreasing a model’s sensitivity to small perturbations, current defenses simultaneously *increase* a model’s vulnerability to a different class of attacks, called invariance attacks [120]. Such an attack applies a *large* perturbation that changes an input’s human-assigned label, but keeps the model’s prediction unchanged.

We introduce new algorithms to craft invariance attacks of bounded ℓ_p norm, and illustrate the above tradeoff for a standard digit classification task. We show that state-of-the-art robust models disagree with human labelers on many of our crafted invariance-based attacks, and that the disagreement rate is higher the more “robust” a model is.

We further break a *provably-robust* digit classifier [286] with our attack. This defense is certified to have high accuracy (with respect to a digit’s original label) under any perturbation that changes an input’s pixel values by up to 40%. That is, given a handwritten digit ‘9’, the model is guaranteed to predict the class ‘9’ for any image obtained by changing the original digit’s pixel values by at most 40%. Yet, we show that it is possible to build a perturbation within this norm-bound that transforms the image into a digit ‘8’, according to a cohort of human labelers. Overall, we find that the model’s agreement with human labelers on our invariance attacks is no better than chance. Thus, while the defense’s proof of robustness is *mathematically correct*, it does not imply that the model’s predictions are actually in agreement with human perception.

Our takeaway from the first part of this dissertation is that robustness to adversarial examples remains by-and-large an unsolved problem in machine learning today. Despite some initial progress in making models robust to restricted types of perturbations, existing defense techniques are inherently limited in their ability to tackle adversarial inputs in their full generality. As a result, we posit that machine learning models that are deployed alongside human users in an adversarial setting (e.g., for content blocking on the Web) will remain prime targets for adversarial examples in the future.

Part II: Privacy-preserving Machine Learning

The second part of this dissertation is concerned with machine learning algorithms that protect the privacy of their users. In contrast to adversarial robustness—the subject of the first half of this dissertation—the task of protecting privacy enjoys much stronger formal foundations developed over the past decades [66, 85, 91, 92]. We build upon these foundations and introduce techniques and systems for preserving user privacy, while achieving significantly higher performance (either model accuracy, or speed) compared to prior work.

What does it mean for machine learning to be *private*? Using a machine learning system typically requires users to share their data with other parties. For example, the data of multiple users could be aggregated to train a joint model. When the model is trained, users may have to send

their data to a remote service in order to obtain predictions. There are different ways to define user “privacy” in such a context. We distinguish between two orthogonal and complementary notions of privacy that we call respectively *secure computation* and *differential privacy*.

Secure computation refers to a *cryptographic* notion of privacy, which asks that the protocols used to train or deploy a model emulate “ideal” protocols wherein the users only interact with a trusted third party [91]. For example, a user could receive predictions from a remote service without the remote service provider learning *anything* about the user’s data. In addition, multiple users could securely train a joint model without an adversary learning anything about each user’s data, other than what can be inferred from the output of the protocol itself (i.e., the trained model). Yet, trained models do actually leak a lot of information about individual users’ data [31, 32, 81, 235]. Thus, secure computation is not a sufficiently strong notion of user privacy for training machine learning models.

Differential privacy [66], in turn, asks that whatever an adversary can infer about a user’s data, it could have also inferred *even if the user had not participated in the protocol*. Differential privacy is a very useful notion of privacy for training machine learning models: it permits learning generalizable facts about a population (e.g., how to detect cancer cells in a lung scan), but prevents the inference of individual-level facts (e.g., John Doe’s cancer scans were used to train the model) [66].

Prior work has shown how to use differential privacy to train neural networks that do not leak individual users’ data [1, 198, 234], and how to use secure computation to allow users to privately receive predictions from a trained model [87, 88, 126, 174]. However, in both cases, privacy comes at a heavy cost. Differentially private training results in large drops in model accuracy, whereas protocols for secure computation incur high communication and computation costs.

In the second part of this dissertation, we introduce new techniques for differentially private training that significantly improve model accuracy compared to prior work. We further describe Slalom, a system for secure outsourcing of neural network predictions that is orders-of-magnitude faster than prior systems.

Differentially private learning with better features. Training deep neural networks with differential privacy comes at a significant cost in utility [1, 10, 76, 284]. In fact, on many machine learning benchmarks the accuracy of private deep learning still falls short of “shallow” (non-private) techniques that rely on handcrafted features [52, 187, 188]. This leads to the central question we address in Chapter 6:

Can differentially private learning benefit from handcrafted features?

We answer this question affirmatively by introducing simple and strong handcrafted baselines for differentially private learning. For example, on the CIFAR-10 benchmark [141] we exceed the best accuracy reported in prior work while simultaneously improving the provable privacy guarantee by

130 \times . Our results show that private deep learning remains outperformed by handcrafted priors on many tasks, and thus has yet to reach its “*AlexNet moment*” [142].

Considering these results, which additional resources may we need in order to outperform our handcrafted baselines? We first investigate whether collecting more private training data could help. Indeed, protecting a user’s privacy becomes *easier* as the size of the training data increases, as each user’s relative contribution to the final model is decreased. Thus, given a large enough private training set, we should expect deep learning to outperform our handcrafted baselines. For CIFAR-10, we find that about *an order of magnitude* additional private training data is needed, before end-to-end deep models outperform our handcrafted features baselines.

Second, we consider complementing a private training set with a larger *public* dataset from a similar domain. For example, suppose that we want to train a model on medical scans from hospital patients. Instead of privately training a deep neural network from scratch, we could leverage image features extracted from large public image datasets such as ImageNet [61]. This process is commonly known as *transfer learning* [210]. While differentially private transfer learning has been studied in prior work [1, 197], we find that its privacy-utility guarantees have been severely underestimated. We revisit these results and show that with transfer learning, strong privacy can come at only a minor cost in accuracy.

Overall, we demonstrate that higher quality features—whether handcrafted or transferred from public data—are of paramount importance for improving the performance of private classifiers.

Slalom: Faster private inference with trusted hardware. Suppose that we now want to deploy a trained model to the cloud, so that users can request predictions on their data. This again requires users to sacrifice their privacy: the user’s input and the model’s output are shared with the cloud provider. Users further have to trust the cloud provider with the *integrity* of the computation. That is, a user cannot verify whether the obtained model output was computed correctly or not.

The literature on secure computation offers a solution to this problem [85, 91, 199, 279]. A user and the cloud provider can engage in a cryptographic protocol that results in the user obtaining the model output (along with a proof of correct computation), without the cloud provider learning anything about the user’s input [87, 88, 126, 174]. Unfortunately, these cryptographic protocols incur very high communication or computation costs, and are three to four orders-of-magnitude slower than a non-private baseline.

Trusted Execution Environments (TEEs) offer a more pragmatic solution to our problem [111, 183]. TEEs use hardware and software protections to isolate sensitive code, while *attesting* to its correct execution. By evaluating a machine learning model in a TEE, the cloud provider can guarantee privacy and integrity to its remote clients. Existing TEE solutions outperform pure cryptographic approaches by multiple orders of magnitude, but still come at a steep price in performance compared to a non-private baseline (due to the TEE’s computation overhead). This leads us to the main question we address in Chapter 7:

How can we most efficiently leverage TEEs for secure machine learning?

We introduce a new approach to this problem, wherein a neural network execution is *partially outsourced from a TEE to a co-located, untrusted but faster device*. The main observation that guides our approach is that matrix multiplication—the computational bottleneck in modern neural networks—admits a concretely efficient verifiable outsourcing scheme known as Freivalds’ algorithm [82], which can also be turned private in our setting. Our TEE selectively outsources computationally intensive steps to a fast untrusted co-processor (e.g., a GPU), thereby achieving much better performance than running the entire computation in the TEE—without compromising security.

We instantiate this approach in Slalom, a system for efficient neural network inference in any trusted execution environment. We implement a Slalom prototype using Intel SGX [164], and evaluate our system on canonical neural networks with a variety of computational costs. Compared to running all computations in the TEE, Slalom increases throughput (as well as energy efficiency) by $6\times$ to $20\times$ for verifiable inference, and by $4\times$ to $11\times$ for verifiable and private inference.

Taken together, the results in the second part of this dissertation demonstrate that it is possible to achieve strong privacy guarantees for machine learning, along with much lower performance overheads than in prior work.

1.2 Machine Learning Background

Most of this dissertation focuses on standard classification tasks, for a distribution \mathcal{D} over examples $x \in \mathbb{R}^d$ and labels $y \in [C]$. A classifier is a function $f_\theta : \mathbb{R}^d \rightarrow [C]$. This function is parametrized by parameters (or weights) $\theta \in \mathbb{R}^p$, which we usually ignore for notational convenience.

The performance of a classifier f on a labeled input (x, y) is measured by means of a loss function $L(x, y; f)$, for example the “0/1” loss or misclassification loss:

$$L_{0/1}(x, y; f) := \mathbb{1}_{\{f(x) \neq y\}} = \begin{cases} 1 & \text{if } f(x) \neq y \\ 0 & \text{otherwise .} \end{cases} \quad (1.1)$$

We will drop the dependency on the classifier f when it is clear from context and write $L(x, y)$.

The *risk* of a classifier, $\mathcal{R}(f)$ is the expected 0/1 loss over the distribution \mathcal{D} :

$$\mathcal{R}(f) := \mathbb{E}_{(x,y) \sim \mathcal{D}} [L_{0/1}(x, y; f)] = \Pr_{(x,y) \sim \mathcal{D}} [f(x) \neq y] . \quad (1.2)$$

Given a dataset $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$ sampled i.i.d. from the distribution \mathcal{D} , and a

loss function L , we define the *empirical risk* $\hat{R}(f)$ as:

$$\hat{R}(f) := \frac{1}{n} \sum_{i=1}^n L(x^{(i)}, y^{(i)}; f). \quad (1.3)$$

A classifier f can be trained with *Empirical Risk Minimization* (ERM), which consists in finding parameters θ^* for f that minimize the empirical risk $\hat{R}(f)$. For this purpose, the loss function L is typically chosen as a differentiable approximation to the 0/1 loss, e.g., the cross-entropy loss.

Neural networks. Most of the classifiers $f(x)$ that we consider are *neural networks*. A neural network is a function $F : \mathbb{R}^d \rightarrow \mathbb{R}^C$ that takes an input $x \in \mathbb{R}^d$ and produces an output vector in \mathbb{R}^C . The network’s output vector corresponds to confidence scores for individual classes. The classifier $f : \mathbb{R}^d \rightarrow [C]$ is defined as $f(x) = \arg \max_{i \in [C]} F(x)_i$.

A neural network F consists of multiple layers:

$$F(x) := F_n \circ F_{n-1} \circ \dots \circ F_1(x).$$

For a standard feed-forward network, each of the n individual layers is of the form

$$F_i(x) := \sigma(w_i \cdot x + b_i),$$

where $w_i \cdot x + b_i$ is some linear operation of the layer’s input x (e.g., a convolution) parameterized by weights w_i and biases b_i , and σ is an activation function (e.g., the ReLU function [178]). The collections of weights $\{w_i\}_{i=1}^n$ and biases $\{b_i\}_{i=1}^n$ make up the network’s parameters θ .

Image classification. While many of the techniques in this dissertation are agnostic to the particular application area for neural networks, we usually illustrate our results on classical tasks from computer vision, in particular image classification.

In an image classification task, the model’s input is an image of dimension $h \times w \times c$. The channel dimension c is equal to 1 for grayscale images, to 3 for RGB images, and to 4 for RGBA images with an extra alpha channel. The inputs are normalized so that each pixel lies in the range $[0, 1]$. An input x thus lies in the range $x \in [0, 1]^{h \cdot w \cdot c}$. The classifier’s output range $[C]$ corresponds to a set of predefined object labels (e.g., $1 \rightarrow$ “horse”, $2 \rightarrow$ “car”, etc.)

Bibliographic Notes

The material in this dissertation is based on the following peer-reviewed works:

Chapter 3: “Ad-versarial: Perceptual Ad-Blocking meets Adversarial Machine Learning”, with Pascal Dupré, Gili Rusak, Giancarlo Pellegrino and Dan Boneh, published in the Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), 2019 [256].

Chapter 4: “Adversarial Training and Robustness for Multiple Perturbations”, with Dan Boneh, published in Advances in Neural Information Processing Systems (NeurIPS), 2019 [250].

Chapter 5: “Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations”, with Jens Behrmann, Nicholas Carlini, Nicolas Papernot and Jörn-Henrik Jacobsen, published in the Proceedings of the International Conference on Machine Learning (ICML), 2020 [257].

Chapter 6: “Differentially Private Learning Needs Better Features (or Much More Data)”, with Dan Boneh, published in the Proceedings of the International Conference on Learning Representations (ICLR), 2021 [252].

Chapter 7: “Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware”, with Dan Boneh, published in the Proceedings of the International Conference on Learning Representations (ICLR), 2019 [251].

Additional Resources

For each of the above works, we further release open-source code and data to reproduce our results:

Chapter 3: <https://github.com/ftramer/ad-versarial>.

Chapter 4: <https://github.com/ftramer/MultiRobustness>.

Chapter 5: <https://github.com/ftramer/Excessive-Invariance>.

Chapter 6: <https://github.com/ftramer/Handcrafted-DP>.

Chapter 7: <https://github.com/ftramer/slalom>.

Notation

We use the following notation throughout this dissertation.

Sets, Algebra and Logic. For a set S , the notation $x \leftarrow S$ indicates that the element x is sampled uniformly at random from S . For an integer $n > 0$, we use the notation $[C] = \{1, \dots, n\}$. When defining a variable X , we use the notation $X := \text{expression}$. We denote the reals by \mathbb{R} , and a finite field by \mathbb{F} . Given a predicate P , we use the notation $\mathbb{1}_{\{P\}}$ for an indicator function that equals 1 if the predicate P is true, and 0 otherwise.

Probability. We denote distributions using calligraphic letters, e.g., $\mathcal{D}, \mathcal{P}, \mathcal{Q}$. We use $x \sim \mathcal{D}$ to denote sampling an element x from the distribution \mathcal{D} . For a random variable X and event O , we denote the probability of the event as $\Pr[X = O]$, the expectation of X as $\mathbb{E}[X]$ and its variance as $\text{Var}[X]$. We denote the normal distribution with mean μ and standard deviation σ as $\mathcal{N}(\mu, \sigma^2)$.

Vectors. We denote real-valued vectors as $x = (x_1, \dots, x_n)$. An indexed of vectors is denoted as $\{x^{(1)}, \dots, x^{(n)}\}$. We will use multiple vector norms throughout this dissertation. The ℓ_∞ norm is defined as $\|x\|_\infty := \max_{1 \leq i \leq n} |x_i|$. The ℓ_2 norm is given by $\|x\|_2 := (\sum_{i=1}^n (x_i)^2)^{1/2}$. The ℓ_1 norm is defined as $\|x\|_1 := \sum_{i=1}^n |x_i|$. The ℓ_0 “norm” (which is not a norm) is the number of non-zero elements of x , i.e., $\|x\|_0 := \sum_{i=1}^n (x_i)^0$ (where we use the convention that $0^0 = 0$). We sometimes use the notation $|x|$ for the number of elements in a vector, i.e, $|x| = n$ in the examples above.

Machine learning. We consider standard supervised classification tasks for a distribution \mathcal{D} over examples $x \in \mathbb{R}^d$ and labels $y \in [C]$. A classifier is a function $f : \mathbb{R}^d \rightarrow [C]$. We denote a dataset sampled from the distribution \mathcal{D} as $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$. We use the notation $L(x, y; f)$ (or $L(x, y)$ when f is clear from context) to denote a loss function applied to the classifier’s output and the input’s true label.

Adversarial perturbations. For an input x , we denote an adversarial version of the input as $\hat{x} := x + \delta$, where δ is an additive perturbation. Given a loss function $L(x + \delta, y; f)$ over a perturbed input, we denote the gradient of L with respect to the perturbation δ as $\nabla_\delta L(x + \delta, y; f)$.

Algorithms. We use the notation $x \leftarrow v$ to indicate assignment of value v to a variable x .

Part I

The Security Threat of Adversarial Examples

In the first part of this dissertation, we study the threat posed by adversarial examples to the security of machine learning systems. In Chapter 2, we formalize the threat model of adversarial examples—which is often left implicit in the literature—and argue that this threat model is unrealistically restrictive for many security-sensitive applications of machine learning. In Chapter 3, we demonstrate a compelling security-relevant application of adversarial examples for evading content moderation systems on the Web. In particular, we show that recent proposals to use machine learning for ad-blocking fail to account for these models’ critical lack of robustness to attacks. We then introduce and analyze two intrinsic limitations of current defenses against adversarial examples, which limit the usefulness of these defenses in practice. In Chapter 4, we show that the performance of current defenses degrades rapidly as we aim for robustness against rich sets of input perturbations. In Chapter 5, we argue that aiming for robustness to well-defined formal sets of perturbations is inherently insufficient, and possibly even harmful. Our main takeaway is that robustness to adversarial examples remains by-and-large an unsolved problem in machine learning today.

Chapter 2

The Threat Model of Adversarial Examples

In this chapter, we introduce a formal security model for adversarial examples, a class of evasion attacks [17] on machine learning models where an adversary tampers with a model’s input to cause the model to fail.

Following a long line of prior work [17, 26, 95, 100, 144, 154, 191, 192, 194, 246], we define an adversarial example for an input x of a classifier f as a perturbed input \hat{x} that is perceptually close to x , and that is misclassified by f :

Definition 2.1 (Adversarial Example). Given a classifier f , an input $(x, y) \sim \mathcal{D}$, and a set S of perceptually small perturbations, an adversarial example for x is an input

$$\hat{x} := x + \delta, \quad \text{such that} \quad f(\hat{x}) \neq y \text{ and } \delta \in S.$$

Under this definition, the problem of designing robust machine learning models can be cast as the minimization of a classifier’s *adversarial risk*:

Definition 2.2 (Adversarial Risk). Given a classifier f , a distribution \mathcal{D} , and a set S of perceptually small perturbations, the adversarial risk $\mathcal{R}_{\text{adv}}(f; S)$ is given by:

$$\mathcal{R}_{\text{adv}}(f; S) := \Pr_{(x, y) \sim \mathcal{D}} \left[\max_{\delta \in S} \mathbb{1}_{\{f(x+\delta) \neq y\}} \right].$$

This formulation is compelling from an optimization perspective [159], and prior work has exploited this to build practical defense techniques that successfully minimize the adversarial risk, both empirically and provably [53, 83, 146, 159, 207, 271].

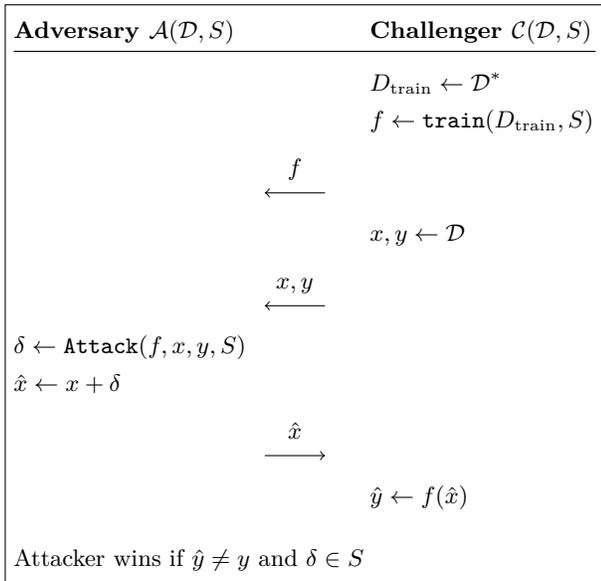


Figure 2.1: **The “expectimax” security game for adversarial examples.** The adversary and challenger know the data distribution \mathcal{D} and the set of perturbations S that a model should be robust to. The challenger first trains a model f and shares this model with the adversary. For a test example (x, y) sampled from the distribution \mathcal{D} , the attacker wins if they find a perturbation $\delta \in S$ such that the perturbed adversarial example \hat{x} is misclassified.

2.1 The Expectimax Game

From a security perspective, the above formulation of adversarial examples and of the adversarial risk leaves implicit a number of important details relating to the actual *threat model* that adversarial examples capture. Instead, we find it helpful to define an analogous *security game*, between an adversary \mathcal{A} and a challenger (or defender) \mathcal{C} , in Figure 2.1.

This game has previously been referred to as the “expectimax” game [93] (because the adversary’s success is given by the *expectation*, over some distribution, of the model’s performance on a *worst-case* perturbation). The adversary’s success in the expectimax game is precisely upper-bounded by the adversarial risk $\mathcal{R}_{\text{adv}}(f; S)$. This upper-bound is tight if the adversary’s attack strategy **Attack** always finds the worst-case adversarial perturbation within the set S .

What threat model does the expectimax game in Figure 2.1 capture? The challenger’s goal is to (robustly) classify inputs from the distribution \mathcal{D} *on average*. The adversary, in turn, has to attack specific examples sampled from this distribution, using perturbations from a pre-defined set. Crucially, the adversary does not get to choose the distribution over inputs, or modify it in any way, beyond perturbing inputs within the pre-defined set S .

This threat model is quite restrictive! For example, it does not allow the adversary to *replay* misclassified examples (a so-called “test set attack” [89]), or to pick arbitrary out-of-distribution

examples. The threat model further implies that it is not enough for the adversary to win only once (or occasionally): a strong adversary should succeed with high probability over the distribution \mathcal{D} .

Is the expectimax game realistic? We argue that in many practical settings considered in prior work, adversarial examples are not a *necessary* attack vector, as the above restrictions do not all apply. A similar argument was previously expressed by Gilmer et al. [89]. Consider the following examples:

1. *Fooling self-driving systems.* Self-driving is often used as a motivating scenario for the safety implications of adversarial examples. Indeed, small perturbations to street signs or markings can fool self-driving models and possibly cause crashes [73, 74, 191].

Yet, adversarial examples poorly capture the threat model that self-driving systems operate in. First, average-case success over a fixed distribution of inputs \mathcal{D} and perturbation set S is a poor metric when human lives are at stake. We should expect a safe model to handle out-of-distribution anomalies, even if these are not “perceptually close” to an in-distribution sample (e.g., a STOP sign could have been broken due to bad weather [185]). Second, a motivated malicious party has no reason to limit themselves to perceptually small attacks. An attacker could paint over a street sign, or hold a (real) STOP sign outside of their car window on the highway. In both cases, we should expect a self-driving system to still operate reliably.

2. *Inaudible audio commands.* Adversarial examples for audio recognition systems have also received a lot of attention [28, 29, 50]. Here, the goal of an attacker is to minimally distort a given audio sample so as to fool the recognition system into hearing audio of the attacker’s choosing (a *targeted* attack). Such an attack can cause a voice assistant to hear commands that are inconspicuous to the owner (and e.g., cause the assistant to issue an unwanted purchase).

The expectimax threat model fails to capture many relevant attack vectors in this setting. For example, a single command that fools the voice assistant could be replayed many times. Another concerning threat is that a perfectly audible command could trigger a voice assistant without the owner’s awareness (e.g., some TV ads mistakenly trigger Alexa devices, and this may happen when the owner is in a different room). This example illustrates that for voice assistants, the problem of *command authenticity* is as much of a concern as *reliable transcription*. The threat model of adversarial examples only considers the latter.

3. *Evading facial recognition.* Adversarial examples have also been proposed as a means to evade facial recognition systems, e.g., by means of small perturbations printed on glasses [228].

Whether “small” perturbations are needed in this scenario is debatable. One could evade a facial recognition system by wearing a face mask (a “perturbation” that is clearly perceptible). But such a subterfuge might be easily detected by a human observer. Yet, researchers have

also shown how to successfully bypass facial recognition systems in airports (where human operators are present) using prosthetic masks—which are also “large” perturbations [223].

The above examples show that the threat model of (minimally perturbed, in distribution) adversarial examples is poorly aligned with the true threat model in which many machine learning models operate. Note that this does not mean that the above models are therefore easier to defend against attack. Quite the opposite in fact: these applications face threats *far broader* than imperceptible perturbations. As a result, Gilmer et al. [89] have argued that the expectimax threat model in Figure 2.1 is not relevant in *any* practical security-sensitive application.

2.2 Adversarial Examples As a Necessary Attack Vector

In Chapter 3, we describe a security-sensitive application where adversarial examples are *necessary* for a successful attack: online ad-blocking.

In this setting, the challenger is an ad-block provider that builds a model that users run in their browsers to detect and block online ads in real time. The adversary (a website publisher or an ad network) aims to show ads to end users by bypassing automated blocking. The threat model faced by online ad-blockers perfectly aligns with the expectimax threat model:

- There is a natural distribution over inputs that the adversary cannot arbitrarily control. Indeed, ads are designed by marketing teams to promote specific products and maximize user engagement. Even if a specific ad bypasses detection, an ad network cannot simply replay that ad indefinitely.
- Perturbations applied to ads (or other web content) should be imperceptible. The goal of the adversary is to show the original ads to users so that users interact with them.
- Average-case error is a reasonable performance metric. Any individual failure on the side of the challenger (i.e., the ad-blocker missing an ad, or incorrectly blocking non-ad content) has a small utility cost. An ad-blocker that blocks 99% of ads would likely be acceptable for most users. In turn, any successful attack from an ad-network or website publisher yields a fixed monetary gain (the expected gain from an ad impression).

The above threat model is not unique to ad-blocking. It applies broadly to the problem of online content blocking or moderation. Other prominent examples include the detection of inappropriate content on social media platforms (where the adversary aims to publish some piece of media with minimal changes so as to bypass detection), or the detection of phishing websites (where the adversary creates a website that visually mimics some authoritative website to trick users).

In Chapter 3, we focus on adversarial examples in ad-blocking for two compelling reasons:

1. Recent progress in computer vision has motivated the design of *perceptual* ad-blockers, that use computer vision techniques to mimic the way in which end users detect ads using visual cues [243, 244, 261, 272]. Perceptual ad-blocking holds the promise of being more robust than classical approaches to ad-blocking based on ad metadata, but its robustness had not been previously evaluated.
2. Ad-blockers operate *client-side*, and are thus typically accessible to the adversary as a *white box*. This makes it considerably easier for an adversary to craft adversarial examples.

We show that perceptual ad-blockers are easily bypassed by exploiting adversarial examples in the underlying computer vision system. Using standard attack techniques, the ad-blocker’s adversaries can craft imperceptible perturbations for ads or other website content that cause the ad-blocker to miss ads. Specifically, the adversary finds perturbations δ from within some imperceptible set S that can be applied to various web elements and that cause the ad-blocker to misclassify the perturbed content. Our attacks apply broadly to different computer vision techniques for ad-blocking, from classical template matching algorithms that detect ad identifiers in individual HTML images [244], to end-to-end neural networks that operate over rendered web content [184, 261].

The fate of perceptual ad-blockers thus appears intimately tied to our ability to design robust visual classifiers, under the expectimax threat model. As we show in Chapter 4 and Chapter 5, there remain fundamental and substantial obstacles towards this goal. These obstacles are intimately tied to the problem of choosing an appropriate perturbation set S in the expectimax game.

2.3 Choosing a Perturbation Set

There is one crucial detail that the game in Figure 2.1 leaves ill-defined: how do we characterize the set S of “small” perturbations? For any input x and perturbation δ , we could of course ask a cohort of human whether $x + \delta$ is perceptually similar to x . Yet, efficiently finding worst-case perturbations from this implicitly defined set seems intractable.

Prior work has sidestepped this issue by choosing a simple “proxy” set $S' \subset S$, such that S' can be efficiently optimized over [95]. Robustness to perturbations from S' is a *necessary* condition for robustness to the entire set S . A popular choice has been to set S' as some small ℓ_p ball, i.e., all perturbations δ such that $\|\delta\|_p$ is below some threshold ε [95, 144, 159, 255].

Focusing on a proxy set S' is sufficient when designing *attacks*. Indeed, if the adversary \mathcal{A} succeeds in finding an adversarial perturbation $\delta \in S'$, then we have $\delta \in S$ as well. An analogous argument does not hold for defenses, and this is the main reason why defending against adversarial examples is so much harder than attacking. Indeed, if a model is robust against perturbations from S' , this does not imply that it is robust against all perturbations in S . In fact, prior work has shown that given two different simple proxy sets (e.g., perturbations of small ℓ_∞ and small ℓ_1 norms

respectively), defenses that are tailored to be robust against perturbations from one of the two sets attain no robustness to perturbations from the other [71, 230].

In Chapter 4, we ask whether it is possible to build robust defenses by approximating S in a “bottom-up” fashion. That is, given multiple proxy sets $S'_i \subset S$, can we train models that are robust against perturbations from the *union* of these proxy sets, $\cup_i S'_i \subset S$. We first prove that aiming for robustness against multiple perturbation types can lead to a *robustness tradeoff*, where increasing robustness to one type of perturbation necessarily decreases robustness to another. We call such perturbation types *mutually exclusive*. We then demonstrate empirically that this robustness tradeoff does manifest itself when training robust vision models against a union of just two or three proxy perturbation types. Thus, this explicit approach of building up robustness to perturbations from increasingly larger subsets of S does not appear promising.

In Chapter 5, we further ask whether the implicit assumption that $S' \subset S$ always holds, i.e., whether the perturbations in the proxy set are truly perceptually small. We find that for some canonical vision benchmarks, the proxy sets considered in the literature are *too large*, and include perturbations that can change an image’s class according to human labelers. Defenses that aim for robustness against these proxy sets thus *harm* a model’s (true) robustness, by making the model invariant to real semantics of the classification task [120]. This situation may seem relatively benign. Yet, we prove that there is a fundamental tradeoff at play here. Informally, unless the “geometry” of the proxy set S' is perfectly aligned with that of the set S , the proxy set S' must either be too small (and thus the defense remains overly sensitive to small perturbations) or too large (in which case the defense is overly invariant to task-relevant features). Unfortunately, we show that finding a proxy set S' with the right “geometry” is as hard as perfectly solving the classification task.

We are thus left with a “chicken-and-egg” problem. If we cannot approximate the set S of perceptually small perturbations, then it seems hard to (explicitly) build models that are robust to perturbations in S . In turn, explicitly approximating S would defeat the purpose of machine learning, which is to *learn* a model of perceptual similarity from data, instead of having to formally characterize it. Our main hope then is that a machine learning model would *implicitly* learn the right notion of perceptual similarity—when given sufficiently rich training data. While this endeavor has been unsuccessful so far, OpenAI’s recent work on internet-scale multimodal learning (CLIP) [206] showed promising improvements for weaker forms of model robustness (for average-case, rather than worst-case perturbations).

Chapter 3

A Security Application: Evading Perceptual Ad-blockers

As we have argued in Chapter 2, adversarial examples are often “overkill” when analyzing the security of deployed machine learning systems. Typically, there exist simpler and arguably more natural attack vectors that threaten the system’s integrity. In this chapter, we thus ask:

Is there a security-sensitive task where adversarial examples are a necessary attack vector?

We provide evidence of a security application where imperceptible perturbations to in-distribution examples are indeed necessary for a successful attack: online ad-blocking.

Online advertising is a contentious facet of the Web. Online ads generate over \$200 billion in value [265], but many Internet users perceive them as intrusive or malicious [138, 151, 205, 276]. The growing use of ad-blockers such as Adblock Plus¹ and uBlock² has sparked a fierce arms race with publishers and advertising networks. Current ad-blockers maintain large crowdsourced lists of ad *metadata*—such as page markup and URLs. In turn, publishers and ad networks (including Facebook [167, 260] and 30% of the Alexa top-10K list [290]) continuously adapt and deploy small changes to web page code in an effort to evade, or detect ad-blocking.

Towards visual ad-blocking. This arms race has prompted ad-blockers to search for more robust signals within ads’ visual *content*, as altering these would affect user experience. One such signal is the presence of ad-disclosures such as a “Sponsored” caption or the AdChoices logo [62]), which many ad-networks add for transparency [62]. Storey et al. [244] proposed Ad-Highlighter [243], the first *perceptual ad-blocker* that detects ad-disclosures by combining web-filtering rules and computer vision techniques. Motivated by the alleged superior robustness of perceptual techniques [244],

¹<https://adblockplus.org>. Accessed 2021-6-22.

²<https://www.ublock.org>. Accessed 2021-6-22.

popular ad-blockers now incorporate similar ideas. For example, Adblock Plus supports image-matching filters [203], while uBlock crawls Facebook posts in search for “Sponsored” captions [260].

However, as proposed perceptual ad-blockers still partially use markup as a proxy for ads’ visual content, they appear insufficient to end the ad-blocking arms race. For example, Facebook routinely evades uBlock Origin using increasingly complex *HTML obfuscation* for the “Sponsored” captions (see [260]). Ad-Highlighter’s computer vision pipeline is also vulnerable to markup tricks such as image fragmentation or spriting (see Figure 3.13). Escaping the arms race over markup obfuscation requires perceptual ad-blockers to move towards operating on *rendered* web content. This is exemplified in Adblock Plus’ Sentinel project [272], that uses deep learning to detect ads directly in web page screenshots. On a similar note, Percival [261] is a recently proposed ad-blocker that adds a deep learning ad-classifier into the rendering pipeline of the Chromium and Brave browsers. While these approaches might bring an end to the current markup-level arms race, our results show that visual ad-blocking merely replaces this arms race with a new one, involving powerful attacks that directly target the ad-blockers’ visual classifier.

Adversarial examples for ad-classifiers. The main threat to visual ad-blockers are adversarial examples, which challenge the core assumption that ML can emulate humans’ visual ad-detection. To our knowledge, our attacks are the first application of adversarial examples to a real-world web-security problem.

We rigorously assess the threat of adversarial examples on *seven* visual ad-classifiers: Two computer-vision algorithms (perceptual hashing and OCR) used in Ad-Highlighter [244]; the ad-classification neural networks used by Percival [261] and [112]; a canonical feature matching model based on the *Scale-Invariant Feature Transform* (SIFT) [156]; and two object detector networks emulating Sentinel [272]. For each model, we create imperceptibly perturbed ads, ad-disclosure or native content, that either evade the model’s detection or falsely trigger it (as a means of detecting ad-blocking).

Among our contributions is a new evasion attack [114, 218] on SIFT [156] that is conceptually simpler than prior work [110].

Attacking perceptual ad-blockers such as Sentinel [272] presents the most interesting challenges. For these, the classifier’s input is a screenshot of a web page with contents controlled by different entities (e.g, publishers and ad networks). Adversarial perturbations must thus be encoded into HTML elements that the adversary controls, be robust to content changes from other parties, and scale to thousands of pages and ads. We tackle the adversary’s uncertainty about other parties’ page contents by adapting techniques used for creating physical adversarial examples [7, 228]. We also propose a novel application of *universal adversarial examples* [175] to create a *single* perturbation that can be applied at scale to all combinations of websites and ads with near 100% success probability.

We further show that adversarial examples enable new attacks, wherein malicious content from

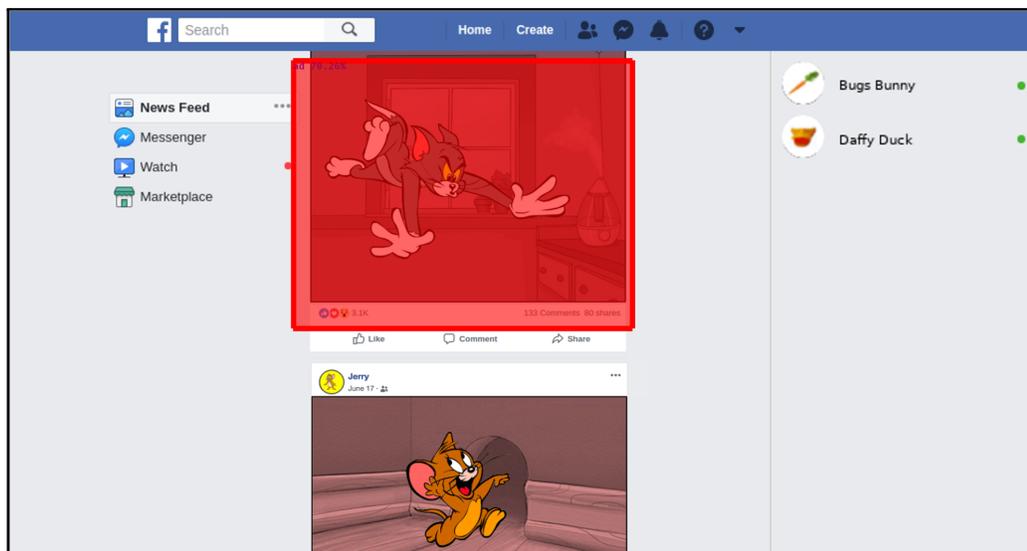


Figure 3.1: **Ad-blocker privilege hijacking.** A malicious user, Jerry, posts adversarial content to Facebook that fools a perceptual ad-blocker similar to Sentinel [272] into marking Tom’s benign content as an ad (red box) and blocking it in every user’s browser.

one user can hijack the ad-blocker’s high privilege to incorrectly block another user’s content. An example is shown in Figure 3.1. Here Jerry, the adversary, uploads a perturbed image to Facebook. That image is placed next to Tom’s post, and confuses the ad-blocker into classifying Tom’s benign post as an ad, and incorrectly blocking it. Hence, a malicious post by one user can cause another user’s post to get blocked.

Moving beyond the Web and visual domain, we build imperceptible audio adversarial examples for AdblockRadio³, a radio ad-blocker that uses ML to detect ads in raw audio streams.

Outlook. While visual ad-classification of rendered web content is both sufficient and necessary to bring an end to the arms race around page markup obfuscation, we show that this merely replaces one arms race with a new one centered on adversarial examples. Our attacks are not just a first step in this new arms race, where ad-blockers can easily regain the upper hand. Instead, they describe an inherent difficulty with the perceptual ad-blocking approach, as ad-blockers operate in essentially the *worst threat model for visual classifiers*. Their adversaries prepare (offline) digital attacks to evade or falsely trigger a known *white-box* visual classifier running inside the ad-blocker. In contrast, the ad-blocker must resist these attacks while operating under strict real-time constraints.

Our study’s goal is not to downplay the merits of ad-blocking, nor discredit the perceptual ad-blocking philosophy. Indeed, ML might one day achieve human-level perception. Instead, we highlight and raise awareness of the inherent vulnerabilities that arise from instantiating perceptual

³<https://www.adblockradio.com>. Accessed 2021-6-22.

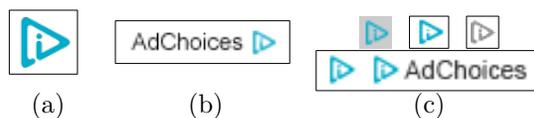


Figure 3.2: **The AdChoices logo.** AdChoices is a standard for disclosure of behavioral advertising [62]. Ads are marked by the icon (a), with optional text (b). Despite creative guidelines [63], many variants of the logo are in use (c).

ad-blockers with existing ML techniques.

3.1 Preliminaries and Background

3.1.1 The Online Advertising Ecosystem

Online advertising comprises four actors: users, publishers, ad networks, and advertisers. Users browse websites owned or curated by a publisher. Publishers assigns parts of the site’s layout to advertisements. Control of these spaces is often outsourced to an ad network that populates them with advertisers’ contents.

To protect users from deceptive ads, the Federal Trade Commission and similar non-US agencies require ads to be clearly recognizable [69]. These provisions have also spawned industry self-regulation, such as the *AdChoices* standard [62] (see Figure 3.2).

3.1.2 Perceptual Ad-blocking

Perceptual ad-blocking aims at identifying ads from their content, rather than from ad *metadata* such as URLs and markup. The insight of Storey et al. [244] is that many ads are explicitly marked—e.g., via a “Sponsored” link or the AdChoices logo—to comply with regulations on deceptive advertising. They developed Ad-Highlighter [243], an ad-blocker that detects ad-disclosures using different perceptual techniques: (i) textual searches for “Sponsored” tags, (ii) fuzzy image search and OCR to detect the AdChoices logo, and (iii) “behavioral” detection of ad-disclosures by identifying links to ad-policy pages. Ad-blockers that rely on perceptual signals are presumed to be less prone to an arms race, as altering these signals would affect user experience or violate ad-disclosure regulations [244].

Perceptual ad-blocking has drawn the attention of major ad-blockers, that have integrated visual signals into their pipelines. For example, uBlock blocks Facebook ads by detecting the “Sponsored” caption. Adblock Plus has added support for image-matching rules, which are easily extended to fuzzy image search [203].

The above perceptual ad-blocking approaches still rely on some markup data as a proxy for ads’ visual content. This has prompted an ongoing arms race between Facebook and uBlock (see [260]) where the former continuously obfuscates the HTML tags that render its “Sponsored” tag—a process

that is invisible to the user. This weakness is fundamental to perceptual approaches that rely on signals with an indirect correspondence to ads’ rendered content. This insight led Adblock Plus to announce the ambitious goal of detecting ads directly from rendered web pages—with no reliance on markup—by leveraging advances in image classification. Their Sentinel [272] project uses an object-detection neural network to locate ads in raw Facebook screenshots. The recently released Percival project [261] targets a similar goal, by embedding a deep-learning based ad-blocker directly into Chromium’s rendering engine.

Design and goals. Ad-blockers are client-side programs running in browsers at a high privilege level. They can be implemented as browser extensions or integrated in the browser. We ignore DNS ad-blockers (e.g., Pi-hole) as these cannot use perceptual signals.⁴

The goal of ad-blockers is to identify and hide ads, while guarding against website breakage resulting from the removal of functional content. As opposed to network-level filters, perceptual signals only apply to downloaded web content and are thus unsuitable for some secondary goals of ad-blockers, such as bandwidth saving or blocking of user tracking and malvertising [138, 151, 205, 276].

Ad-blockers may strive to remove ads without being detected by the publisher. For example, many websites try to detect ad-blockers [176] and take according action (e.g., by asking users to disable ad-blockers). As perceptual ad-blockers do not interfere with web requests, they are undetectable by the web-server [244]. However, the publisher’s JavaScript code can try to detect ad-blockers by observing changes in the DOM page when hiding ads.

Finally, perceptual ad-blockers have strict timing constraints, and should process a web page and detect ads in close to real-time.

Algorithms for visual ad classification. The identification of ads or ad-disclosures can be achieved using a variety of computer vision techniques. Below, we describe existing approaches.

- *Template matching.* Ad-Highlighter detects the AdChoices logo by comparing each image in a page to a template using *average hashing*: for each image, a hash is produced by resizing the image to a fixed size and setting the i^{th} bit in the hash to 1 if the i^{th} pixel is above the mean pixel value. An image matches the template if their hashes have a small Hamming distance.

A more robust template matching algorithm is SIFT [156] (Scale Invariant Feature Transform), which creates an image hash from detected “keypoints” (e.g., edges and corners).

- *Optical Character Recognition.* To detect the rendered text inside the AdChoices logo, Ad-Highlighter uses the open-source Tesseract OCR system⁵. Tesseract splits an image into

⁴While we focus on the desktop browser setting, perceptual ad-blocking might also prove useful in the mobile domain. Current mobile ad-blockers are often part of a custom browser, or act as web proxies—an insufficient approach for native apps that prevent proxying using certificate pinning. Instead, a perceptual ad-blocker (potentially with root access) could detect ads directly from app screenshots

⁵<https://github.com/tesseract-ocr/tesseract>. Accessed 2021-6-22.

overlapping frames and transcribes this sequence using a neural network. Ad-Highlighter matches images for which the OCR output has an edit-distance with “AdChoices” below 5.

- *Image Classification.* Albeit not in an ad-blocking context, Hussain et al. [112] have demonstrated that neural networks could be trained to distinguish images of ads from non-ads (without the presence of any explicit ad-disclosures). The Percival project trained a similar neural network to classify image frames in real-time within Chromium’s rendering pipeline [261].
- *Object Detection.* Sentinel [272] detects ads in rendered web pages using an object detector network based on YOLOv3 [213]. The network’s output encodes locations of ads in an image. The YOLOv3 [213] model outputs bounding box coordinates and confidence scores for $B = 10,647$ object predictions, and retains those with confidence above a threshold $\tau = 0.5$.

3.1.3 Threat Model and Adversaries

We adopt the terminology of adversarial ML [195], where the defenders are users of a classifier (the ad-blocker) that its adversaries (e.g., ad networks or publishers) are trying to subvert.

Publishers, ad networks, and advertisers have financial incentives to evade or detect ad-blockers. We assume that publishers and ad networks are rational attackers that abide by regulations on online advertising, and also have incentives to avoid actively harming users or disrupting their browsing experience. As shown in prior work [201, 276], this assumption fails to hold for advertisers, as some have abused ad-networks for distributing malware. We assume that advertisers and content creators (e.g., a Facebook user) may try to actively attack ad-block users or other parties.

As ad-blockers are client-side software, adversaries can download and inspect their code offline. However, we assume that adversaries do not know *a priori* whether a user is running an ad-blocker.

Attacking ad-blockers. The primary adversarial goal of publishers, ad-networks and advertisers is to evade the ad-blocker’s detection and display ads to users. These adversaries may modify the structure and content of web pages or ads to fool the ad-detector.

Alternatively, the ad-blocker’s adversaries may try to detect its presence, to display warnings or deny access to the user. A common strategy (used by 30% of publishers in the Alexa top-10k) adds fake ad-content (honeypots) to a page and uses JavaScript to check if the ads were blocked [290]. This practice leads to an orthogonal arms race on ad-block detection [176, 177, 182].

Adversaries may also try to abuse ad-blockers’ behaviors to degrade their usability (e.g., by intentionally causing site-breakage or slow performance). The viability of such attacks depends on the adversary’s incentives to avoid disrupting ad-block users’ browsing experience (e.g., Facebook adds honeypots to regular user posts to cause site-breakage for ad-block users [260]).

Finally, attackers with no ties to the online advertisement ecosystem may try to hijack an ad-blocker’s high privilege-level in other users’ machines. Such attackers can act as advertisers or

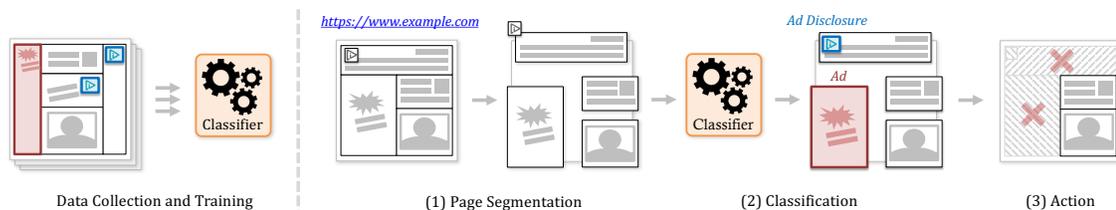


Figure 3.3: **The architecture of a perceptual ad-blocker.** In the offline phase, an ad-classifier is trained on web data. In the online phase, the ad-blocker segments visited pages (1), classifies individual elements (2), and renders the user’s ad-free viewport (3).

content creators to upload malicious content that exploits an ad-blocker’s vulnerabilities. Figure 3.1 shows one example of such an attack, where a malicious Facebook user uploads content that tricks the ad-blocker into hiding an honest user’s posts.

3.2 Designing Perceptual Ad-blockers

To analyze the security of perceptual ad-blockers, we first propose a unified architecture that incorporates and extends prior and concurrent work (e.g., Ad-Highlighter [244], visual filter-lists [203], Sentinel [272], and the recent Percival patch for Chromium’s rendering engine). We explore different ways in which ad-blockers can integrate perceptual signals, and identify a variety of computer vision and ML techniques that can be used to visually identify ads.

To simplify exposition, we restrict our analysis to ad-blockers that only rely on perceptual signals. In practice, these signals are likely to be combined with existing filter lists (as in uBlock [260] or Adblock Plus [203]) but the details of such integrations are orthogonal to our work. We note that an ad-blocker that combines perceptual signals with filter lists inherits the vulnerabilities of both, so our security analysis applies to these hybrid approaches as well.

3.2.1 General Architecture

A perceptual ad-blocker is defined by a collection of offline and online steps, with the goal of creating, maintaining and using a classifier to detect ads. Figure 3.3 shows our unified architecture for perceptual ad-blockers. The ad-blocker’s core visual classifier can range from classical computer vision as in Ad-Highlighter [243] to large ML models as in Sentinel [272].

The classifier may be trained using labeled web data, the type and amount of which varies by classifier. Due to continuous changes in web markup, ad-blockers may need regular updates, which can range from extending existing rules (e.g., for Ad-Highlighter [243, 244]) to re-training complex ML models such as Sentinel [272].

When deployed by a user, the ad-blocker analyzes data from visited pages to detect and block

ads in real-time. Ad detection consists of three main steps. (1) The ad-blocker optionally segments the web page into smaller chunks. (2) A classifier labels each chunk as ad or non-ad content. (3) The ad-blocker acts on the underlying web page based on these predictions (e.g., to remove HTML elements labeled as ads). For some ad-classifiers, the segmentation step may be skipped. For example, Sentinel [272] uses an object-detection network that directly processes full web page screenshots.

Ad-Highlighter’s use of behavioral signals (i.e., recognizing ad-disclosures by the presence of a link to an ad-policy page) can be seen as a special type of classifier that may interact with segmented web elements (e.g., by clicking and following a link).

3.2.2 Approaches to Ad Detection

When online, a perceptual ad-blocker’s first action is the “Page Segmentation” step that prepares inputs for the classifier. Figure 3.4 illustrates different possible segmentations. A cross-origin `iframe` (red box 3) displays an ad and an AdChoices icon (purple box 2). An additional textual ad-disclosure is added by the publisher outside the `iframe` (purple box 1). Publishers may use `iframes` to display native content such as videos (e.g., red box 4).

We distinguish three main perceptual ad-blocking designs that vary in the granularity of their segmentation step, and in turn in the choice of classifier and actions taken to block ads.

- *Element-based perceptual ad-blockers*, such as Ad-Highlighter, search a page’s DOM tree for HTML elements that identify ads, e.g., the AdChoices logo or other ad-disclosures.
- *Page-based perceptual ad-blockers*, e.g., Sentinel [272], ignore the DOM and classify images of rendered web pages.
- *Frame-based perceptual ad-blockers*, e.g., Percival [261], classify rendered content but pre-segment pages into smaller frames.

Element-based perceptual ad-blocking. These ad-blockers segment pages into HTML elements that are likely to contain ad-disclosures. The segmentation can be coarse (e.g., Ad-Highlighter extracts all `img` tags from a page) or use custom filters as in Adblock Plus’ image search [203] or Ublock’s Facebook filters [260].

For textual ad-disclosures (e.g., Facebook’s “Sponsored” tag) the classification step involves trivial string matching. Facebook is thus deploying HTML obfuscation that targets an ad-blocker’s ability to find these tags [260]. This ongoing arms race calls for the use of visual (markup-less) detection techniques. Ad-disclosure logos (e.g., the AdChoices icon) can be visually classified using *template matching*. Yet, due to many small variations in ad-disclosures in use, *exact* matching (as in Adblock Plus [203]) is likely insufficient [244]. Instead, Ad-Highlighter uses *perceptual hashing*

to match all `img` elements against the AdChoices logo. Ad-Highlighter also uses supervised ML—namely Optical Character Recognition (OCR)—to detect the “AdChoices” text [243]. Once an ad-disclosure is identified, the associated ad is found using custom rules (e.g., when Ad-Highlighter finds an AdChoices logo, it blocks the parent `iframe`).

Storey et al. [244] further suggest to detect ads through *behavioral signals* that capture the ways in which users can interact with them, e.g., the presence of a link to an ad-policy page.

Frame-based perceptual ad-blocking. The above element-based approaches require mapping elements in the DOM to rendered content (to ensure that elements are visible, and to map detected ad-identifiers to ads). As we show in Section 3.6.2, this step is non-trivial and exploitable if ad-blockers do not closely emulate the browser’s DOM rendering, a complex process that varies across browsers. For instance, image fragmentation or spriting (see Figure 3.13) are simple obfuscation techniques that fool Ad-Highlighter, and would engender another cat and mouse game. To avoid this, ad-blockers can directly operate on rendered images of a page, which many browsers (e.g., Chrome and Firefox) make available to extensions. Instead of operating on an entire rendered web page (see page-based ad-blockers below), DOM features can still be used to segment a page into regions likely to contain ads. For example, segmenting a page into screenshots of each `iframe` is a good starting point for detecting ads from external ad networks. The approach of Percival is also frame-based but directly relies on image frames produced during the browser’s rendering process [261].

We consider two ways to classify frames. The first searches for ad-disclosures in rendered ads. Template-matching is insufficient due to the variability of backgrounds that ad-disclosures are overlaid on. Instead, we view this as an object-detection problem and address it with supervised ML. The second approach is to train a visual classifier to directly detect ad content. Hussain et al. [112] report promising results for this task. Percival also relies on a lightweight deep learning model to classify frames as ad content [261].

Page-based perceptual ad-blocking. The core idea of perceptual ad-blocking is to emulate the way humans detect ads. Element- and frame-based approaches embrace this goal to some extent, but still rely on DOM information that humans are oblivious to. Recently, Adblock Plus proposed an approach that fully emulates visual detection of online ads from rendered web content alone [272].

In a page-based ad-blocker, segmentation is integrated into the classifier. Its core task is best viewed as an object-detection problem: given a web page screenshot, identify the location and dimension of ads. Adblock Plus trained the YOLOv3 object-detector [213] on screenshots of Facebook with ads labeled using standard filter-lists.

Once ad locations are predicted, the ad-blocker can overlay them to hide ads, or remove the underlying HTML elements (e.g., by using the `document.elementFromPoint` browser API to get the HTML element rendered at some coordinate).

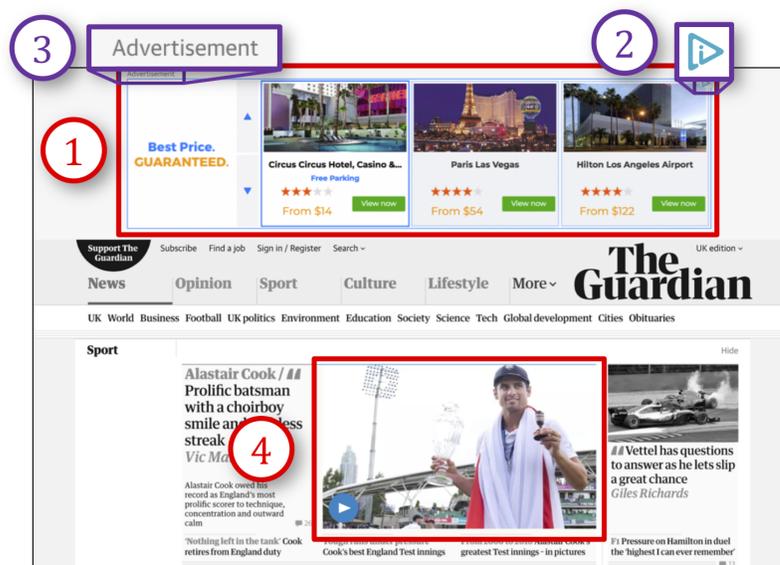


Figure 3.4: **Perceptual ad-blocking elements.** An ad (box #1) is displayed in an `iframe`, that contains an AdChoices icon (box #2). A custom ad-disclosure from the publisher is outside the `iframe` (box #3). Publishers can use `iframes` to display non-ad content such as videos (box #4).

3.3 Training a Page-based Ad-blocker

As the trained neural network of Sentinel [272] is not available for an evaluation, we trained one for the analysis of Section 3.4. We used the same architecture as Sentinel, i.e., YOLO (v3) [212, 213, 214].

3.3.1 Data Collection

YOLO is an object detection network. Given an image, it returns a set of bounding boxes for each detected object. To train and evaluate YOLO, we created a dataset of labeled web page screenshots where each label encodes coordinates and dimensions of an ad on the page. We created the dataset with an ad-hoc automated system that operates in two steps. First, given a URL, it retrieves the web page and identifies the position of ads in the page using filter lists of traditional ad-blockers. Then, our system generates a web page template where ads are replaced with placeholder boxes. The concept of web page templates is convenient as it enables us to create multiple screenshots from the same web page with different ads, a form of data-augmentation. Second, from each web page template, we derive a number of images by placing ads on the template.

Web pages. We acquired web pages by retrieving the URLs of the top 30 news websites of each of the G20 nations listed in `allyoucanread.com`. For each news site, we searched for the RSS feed

URLs and discarded sites with no RSS feeds. The total number of RSS feed URLs is 143. We visited each RSS feed URL daily and fetched the URLs to the daily news.

Template generation. Given a URL of a news article, we generate a page template using a modified HTTP proxy that matches incoming HTTP requests against traditional ad-blocker filter lists, i.e., Easylist⁶ and Ghostery⁷. The proxy replaces ad contents with monochrome boxes using a unique color for each ad. These boxes are placeholders that we use to insert new ads. We manually inspected all templates generated during this step to remove pages with a broken layout (caused by filter lists’ false positives) or pages whose ads are still visible (caused by filter lists’ false negatives).

Image generation. From each page template, we generate multiple images by replacing placeholder boxes with ads. We select ads from the dataset of Hussain et al. [112]. This dataset contains about 64K images of ads of variable sizes and ratios. We complemented the dataset with 136 ads we retrieved online. To insert pictures inside a template, we follow four strategies:

1. We directly replace the placeholder with an ad;
2. We replace the placeholder with an ad, and we also include an AdChoices logo in the top right corner of the ad;
3. We augment templates without placeholders by adding a large ad popup in the page. The page is darkened to highlight the ad;
4. We insert ads as background of the website, that fully cover the left- and right-hand margins of the page.

When inserting an ad, we select an image with a similar aspect ratio. When we cannot find an exact match, we resize the image using Seam Carving [8], a content-aware image resizing algorithm that minimizes image distortion. To avoid overfitting during training, we limited the number of times each ad image can be used to 20.

3.3.2 Evaluation and Results

Datasets. The training set contains 2,901 images, of which 2,600 have ads. 1,600 images with ads were obtained with placeholder replacement, 800 with placeholder replacements with AdChoices logos, 100 with background ads, and 100 with interstitials.

The evaluation set contains 2,684 images—2,585 with ads and 99 without ads. These are 1,595 images with placeholder replacement, 790 images with placeholder replacement with AdChoices logos, 100 images with background ads, and 100 images with interstitials. We also compiled a

⁶<https://easylist.to>. Accessed 2021-6-22.

⁷<https://www.ghostery.com>. Accessed 2021-6-22.

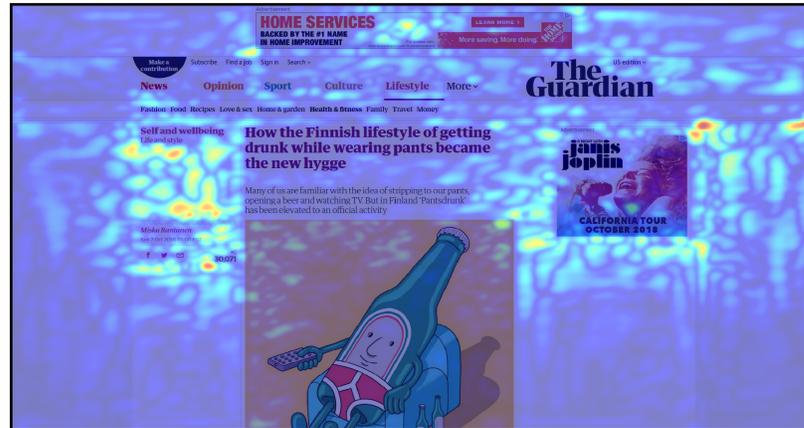


Figure 3.5: **Activation maps of our ad detection model.** The most salient features appear to be the surroundings of ads rather than their visual content.

second evaluation set from 10 domains not used for training (this set is different from the one used to evaluate attacks in Section 3.4). For each domain, we took a screenshot of the front page and four screenshots of different subpages, resulting in 50 screenshots overall with a total of 75 advertisements. We trained using the default configuration of YOLOv3 [213], adapted for a unary classification task.

Accuracy and performance. We tested our model against both evaluation sets. The model achieved the best results after 3,600 training iterations. In the first set, our model achieved a mean average precision of 90.88%, an average intersect of union of 84.23% and an F1-score of 0.96. On the second set, our model achieved a mean average precision of 87.28%, an average intersect of union of 77.37% and an F1-score of 0.85. A video demonstrating our model detecting ads on five never seen websites is available at <https://github.com/ftramer/ad-versarial/blob/master/videos>.

We evaluate performance of the model in TensorFlow 1.8.0 with Intel AVX support. On an Intel Core i7-6700 CPU the prediction for a single image took 650ms.

Inspecting our model. We conduct a preliminary study of the inner-workings of our neural network. By inspecting the model’s *activation map* on different inputs (see Figure 3.5), we find that the model mainly focuses on the layout of ads in a page, rather than their visual content. This shows that our ad-blocker detects ads using very different visual signals than humans. This raises an intriguing question about the Sentinel model of Adblock Plus [272], which was trained solely on Facebook data, where ads are visually close to the website’s native content. Thus, it seems less likely that Sentinel would have learned to detect ads using layout information.

To generate the map in Figure 3.5, we compute the absolute value of the gradient of the network’s output with respect to every input pixel, and apply a smoothing Gaussian kernel over the resulting image. The gradient map is then overlaid on the original input.

Table 3.1: **Attack strategies on perceptual ad-blockers.** Strategies are grouped by the component that they exploit—(D)ata collection, (S)egmentation, (C)lassification, (A)ction. For each strategy, we specify which goals it can achieve, which adversaries can execute it, and which ad-blockers it applies to (fully: ● or partially: ◐).

Strategy	Goals			Actors				Target		
	Evasion	Detection	Abuse	Publisher	Ad Network	Advertiser	Content creator	Element-based	Frame-based	Page-based
D1: Data Training Poisoning	●	●	●	●	●	●	●	●	●	●
S1: DOM Obfuscation	●	●	○	●	●	○	○	●	◐	○
S2: Resource Exhaustion (over-Segmentation)	●	○	○	●	●	◐	◐	●	◐	○
C1: Evasion with Adversarial Ad-Disclosures	●	○	○	○	●	○	○	●	○	○
C2: Evasion with Adversarial Ads	●	○	○	○	●	●	○	○	●	●
C3: Evasion with Adversarial Content	●	○	○	●	○	○	●	○	○	●
C4: Detection with Adversarial Honeypots	○	●	○	●	○	○	○	●	●	●
A1: Cross-Boundary Blocking	○	○	●	○	○	●	●	○	○	●
A2: Cross-Origin Web Requests	○	○	●	○	○	●	●	●	○	○

3.4 Evaluating the Robustness of Perceptual Ad-blocking

Given the unified architecture from Section 3.2, we now perform a comprehensive security analysis of the perceptual ad-blocking pipeline and describe multiple attacks targeting concrete instantiations of each of the ad-blocker’s components. The primary focus of our analysis is to evaluate the robustness of the ad-blocker’s core visual classifier, by instantiating adversarial examples for seven different and varied approaches to ad-detection (Section 3.5). We further demonstrate powerful attacks that exploit the ad-blocker’s high-privilege actions (Section 3.6.1). We conclude by describing more classical attacks that affect the segmentation step of current perceptual ad-blockers (Section 3.6.2), as well as potential attacks on an ad-blocker’s offline data collection and training phase (Section 3.6.3).

Our attacks can be mounted by different adversaries (e.g., publishers, ad-networks, or malicious third parties) to evade or detect ad-blocking and, at times, abuse the ad-blocker’s high privilege level to bypass web security boundaries. These attacks, summarized in Table 3.1, challenge the belief that perceptual signals can tilt the arms race with publishers and ad-networks in favor of ad-blockers.

The attacks described in this section do not violate existing laws or regulations on deceptive advertising, as the changes to the visual content of a page are imperceptible to human users.

Table 3.2: **Evaluation of ad-classifiers.** For each classifier, we first evaluate on “benign” data collected from websites. We report false-positives (FP)—mis-classified non-ad content—and false negatives (FN)—ad-content that the classifier missed. We then give the the attack model(s) considered when *evading* the classifier, the success rate, and the corresponding section.

Category	Method	Targets	Benign Eval.		Adversarial Eval.	
			FP	FN	Attack Model for Evasion	Success
Element-based	Blacklist	AdChoices logos	0/824	33/41	N.A.	-
	Avg hash [244]	AdChoices logos	3/824	3/41	Add ≤ 3 empty rows/cols	100%
	SIFT	textual AdChoices	2/824	0/17	$\ell_2 \leq 1.5$	100%
	OCR [244]	textual AdChoices	0/824	1/17	$\ell_2 \leq 2.0$	100%
Frame-based	YOLOv3	AdChoices in <code>iframe</code>	0/20	5/29	$\ell_\infty \leq 4/255$	100%
	ResNet [112]	ad in <code>iframe</code>	0/20	21/39	$\ell_\infty \leq 2/255$	100%
	Percival [261]	large ads in <code>iframe</code>	2/7	3/33	$\ell_\infty \leq 2/255$	100%
Page-based	YOLOv3	ads visible in page screenshot	2	6/30	Publisher: universal full-page mask (99% transparency) Publisher: adv. content below ads on BBC.com, $\ell_\infty \leq 3/255$ Ad network: universal mask for ads on BBC.com, $\ell_\infty \leq 4/255$	100% 100% 95%

3.4.1 Evaluation Setup

Evaluated approaches. We analyze a variety of techniques to instantiate the different stages of the perceptual ad-blocking pipeline. In particular, we evaluate *seven* distinct approaches to the ad-blocker’s core visual ad-classification step (see Table 3.2). Three are element-based, three frame-based, and one page-based. These seven classifiers are taken from or inspired by prior work. They are: Two computer vision algorithms used in Ad-Highlighter [243, 244] (average hashing and OCR); two ad classifiers, one from Hussain et al. [112] and one used in Percival [261]; a robust feature matcher, SIFT [156]; and two object detector networks—with the same YOLOv3 model [213] as Sentinel [184, 272]—which we trained to detect either ad-disclosures in frames, or ads in a full web page.

For the two object detector models we built, we explicitly separated (i.e., assigned to non-communicating authors) the tasks of (1) data-collection, design and training; and (2) development of attacks, to ensure fair evaluation results. Our first (frame-based) model was trained to detect AdChoices logos that we overlaid in a dataset of 6,320 ads collected by Hussain et al. [112]. We then classify an `iframe` as an ad, if the model detects the AdChoices logo in it.

Our second model emulates the approach of the unreleased Sentinel [184, 272] and was trained to detect ads in arbitrary news websites. This broadens Sentinel’s original scope (which was limited to Facebook)—a decision we made due to difficulties in collecting sufficient training data [184]. The process is described in Section 3.3. A video of our model in action on five websites not seen during training is available at <https://github.com/ftramer/ad-versarial/blob/master/videos>.

Table 3.3: **Evaluation data for adversarial examples.** We collect images, frames and screenshots from the Alexa top ten news websites that use the AdChoices standard (we exclude `news.google.com` and `shutterstock.com` which contain no ads on their front-page). For each page, we extract all images below 50 KB, all iframes, and take two screenshots (the front page and an article) of the user’s viewport, and report the number of visible ads in these.

Website	Images		Iframes			Visible	
	Total	AdChoices	Total	Ads	AdChoices	Ads	
<code>reddit.com</code>	70	2	2	2	2	2	
<code>cnn.com</code>	36	7	7	5	2	3	
<code>nytimes.com</code>	89	4	3	3	3	2	
<code>theguardian.com</code>	75	4	8	3	3	3	
<code>indiatimes.com</code>	125	4	5	5	4	3	
<code>weather.com</code>	144	5	11	7	3	3	
<code>news.yahoo.com</code>	100	5	3	3	2	3	
<code>washingtonpost.com</code>	40	1	5	2	1	3	
<code>foxnews.com</code>	96	5	6	5	4	4	
<code>huffingtonpost.com</code>	90	4	9	4	5 [†]	4	
Total	865	41	59	39	29	30	

[†] One AdChoices logo appears in two rendered iframes laid on top of each other.

Evaluation data. We use real website data to evaluate the accuracy and robustness of the above seven ad-classifiers. We built an evaluation set from the top ten news websites in the Alexa ranking (see Table 3.3).

For each website, we extract the following data:

1. All images smaller than 50KB in the DOM. This data is used to evaluate element-based techniques. We collect 864 images, 41 of which are AdChoices logos (17/41 logos contain the “AdChoices” text in addition to the icon).
2. A screenshot of each `iframe` in the DOM tree, to evaluate frame-based models. We collect 59 frames. Of these, 39 are ads and 29 contain an AdChoices logo. Percival [261] only considers images of dimension at least 100×100 px so we limit it to these.⁸
3. Two screenshots per website (the front-page and an article) taken in Google Chrome on a 1920×1080 display.⁹ These are used to evaluate page-based models. Each screenshot contains 1 or 2 fully visible ads, with 30 ads in total.

⁸Taking a screenshot of an `iframe` is an approximation of how Chromium’s rendering engine segments frames for Percival’s classifier. We verified that our attacks on Percival’s network work when deployed inside the Chromium browser.

⁹We experimentally verified that our attacks on page-based ad-blockers are robust to changes in the user’s viewport. An attacker could also explicitly incorporate multiple browsers and display sizes into its training set to create more robust attacks. Alternatively, the adversary could first detect the type of browser and viewport (properties that are easily and routinely accessed in JavaScript) and then deploy “responsive” attacks tailored to the user’s setting.

For template-matching approaches (perceptual hashing and SIFT) we use the same 12 AdChoices templates as Ad-Highlighter [243].

When describing an ad-blocker’s page segmentation and the corresponding markup obfuscation attacks in Section 3.6.2, we use some data collected on Facebook.com in November 2018. As Facebook continuously and aggressively adapts the obfuscation techniques it uses to target ad-blockers [260], the specific attacks we describe may have changed, which only goes to illustrate the ongoing arms race and need for more robust markup-less ad-blocking techniques.

3.4.2 Accuracy and Performance of ML classifiers.

Table 3.2 reports the accuracy of the seven ad-classifiers on our evaluation data. For completeness, we include a blacklist that marks any image that exactly matches one of the 12 AdChoices logos used in Ad-Highlighter. As posited by Storey et al. [244], this approach is insufficient.

Note that the datasets described above are incomparable. Some ads are not in `iframes`, or have no ad-disclosure, and screenshots only contain images within the current view. Thus, the accuracy of the classifiers is also incomparable. This does not matter, as our aim is not to find the best classifier, but to show that *all* of them are insecure in the stringent attack model of visual ad-blockers.

Overall, element-based approaches have high accuracy but may suffer from some false-positives (i.e., non-ad content classified as ads) that can lead to site-breakage. The frame-based approaches are less accurate but have no false-positives. Finally, our Sentinel-like detector shows promising (albeit imperfect) results that demonstrate the possibility of ad-detection on arbitrary websites.

We measure performance of each classifier on an Intel Core i7-6700 Skylake Quad-Core 3.40GHz. While average hashing and SIFT process all images in a page in less than 4 seconds, OCR is much slower (Ad-Highlighter disables it by default). Our OCR model parses an image in 100 ms, a 14 second delay on some websites. The frame-based classifiers process all `iframes` in 1-7 seconds. Our page-based model processes pages downsized to 416×416 px at 1.5 frames-per-second (on CPU), which may suffice for ad-blocking. The authors of Percival recently demonstrated that an optimized deployment of perceptual ad-blocking with a deep learning classifier incurs only minimal overhead on page rendering (< 200 ms).

3.5 Attacking Ad Classifiers With Adversarial Examples

For perceptual ad-blockers that operate over images (whether on segmented elements as in Ad-Highlighter [243], or rendered content as in Sentinel [272] or Percival [261]), security is contingent on the robustness of the ad-blocker’s visual classifier. *False negatives* result in ads being shown, and *false positives* cause non-ads to be blocked.

Both error types are exploitable using *adversarial examples* [95, 246]—small input perturbations that fool a classifier. Adversarial examples can be used to generate web content that fools the

ad-blocker’s classifier, without affecting a user’s browsing experience.

In this section, we describe and evaluate four concrete types of attacks on the seven visual classifiers we consider: (C1) *adversarial ad-disclosures* that evade detection; (C2) *adversarial ads* that evade detection; (C3) *adversarial non-ad content* that alters the classifier’s output on nearby ads; (C4) *adversarial honeypots* (misclassified non-ad elements, to detect ad-blocking). Our attacks allow adversaries to evade or detect ad-blocking with (near)-100% probability.

3.5.1 Attack Model

We consider adversaries that perturb web content to produce *false-negatives* (to evade ad-blocking) or *false-positives* (honeypots to detect ad-blocking). Each attack targets a single classifier—but is easily extended to multiple models (see Section 3.7).

- *False negative.* To evade ad-blocking, publishers, ad networks or advertisers can perturb any web content they control, but aim to make their attacks imperceptible. We consider perturbations with small ℓ_2 or ℓ_∞ norm (for images with pixels normalized to $[0, 1]$)—a sufficient condition for imperceptibility. An exception to the above are our attacks on average hashing, which is by design invariant to small ℓ_p changes but highly vulnerable to other imperceptible variations. The attack model used for all evasion attacks are summarized in Table 3.2.
- *False positive.* The space of non-disruptive false positive attacks is vast. We focus on one easy-to-deploy attack, that generates near-uniform rectangular blocks that blend into the page’s background yet falsely trigger the ad-detector.

We assume the publisher controls the page’s HTML and CSS, but cannot access the content of ad frames. This content, including the AdChoices logo, is added by the ad network.

Gilmer et al. [89] argue that the typical setting of adversarial examples, where the adversary is restricted to finding imperceptible perturbations for given inputs, is often unrepresentative of actual security threats. Interestingly, the threat model for visual ad classifiers does align perfectly with this setting. The ad-blocker’s adversaries want to evade its classifier for a specific input (e.g., the publisher’s current web page and an advertiser’s latest ad campaign), while ensuring that the users’ browsing experience is unaffected.

3.5.2 Overview of Attack Techniques and Results

For all seven ad-classifiers, we craft imperceptible adversarial perturbations for ad-disclosures, ads and other web content, which can be used by publishers, ad-networks, or advertisers to evade or detect ad-blocking.

Some of our classifiers can be attacked using existing techniques. For example, we show that ad-networks and publishers can use standard gradient-based attacks [26, 144, 159] to create imperceptibly perturbed ads or background content that fool our two frame-based classifiers with 100% success rates (see Figure 3.7). We verify that similar attacks bypass the model used in Percival [261].

Attacking element-based classifiers is less straightforward, as they operate on small images (adversarial examples are presumed to be a consequence of high dimensional data [90]), and some rely on traditional computer vision algorithms (e.g., average hashing or SIFT) for which gradient-based attacks do not apply. Nevertheless, we succeed in creating virtually invisible perturbations for the AdChoices logo, or background honeypot elements, that fool these classifiers (see Figure 3.6). Our attacks on Ad-Highlighter’s OCR network build upon prior work by Song and Shmatikov [238]. For non-parametric algorithms such as SIFT, we propose a new generic attack using black-box optimization [114, 218] (see Section 3.5), that is conceptually simpler than previous attacks [110].

Our most interesting attacks are those that target page-based ad-blockers such as Sentinel [272] (see Figure 3.11, as well as Figure 3.10). Our attacks let publishers create perturbed web content to evade or detect ad-blocking, and let ad-networks perturb ads that evade ad-blocking on the multitude of websites that they are deployed in. These attacks overcome a series of novel constraints.

First, attacks on visual ML classifiers often assume that the adversary controls the full digital image fed to the classifier. This is not the case for page-based ad-blockers, whose input is a screenshot of a web document with content controlled by different actors (e.g., ad networks only control the content of ad frames, while publishers can make arbitrary website changes but cannot alter ads loaded in cross-origin `iframes`). Moreover, neither actor precisely knows what content the other actors will provide. Adversarial examples for page-based ad-blockers thus need to be encoded into the HTML elements that the adversary controls, and must be robust to variations in other page content. We solve this constraint with techniques similar to those used to make physical-world adversarial examples robust to random transformations [73, 143, 228]. We consider multiple tricks to encode a publisher’s perturbations into valid HTML. One attack uses CSS rules to overlay a near-transparent perturbed mask over the full page (Figure 3.11 (b)). To detect ad-blocking, we craft an innocuous page-footer that triggers the ad-blocker (Figure 3.11 (d)). Details on our attacks are in Section 3.5.

A further challenge is the deployment of these attacks at scale, as creating perturbations for every ad and website is intractable. This challenge is exactly addressed by attacks that create *universal adversarial examples* [175]—*single* perturbations that are crafted so as to be effective when applied to most classifier inputs. Universal perturbations were originally presented as a curious consequence of the geometry of ML classifiers [175], and their usefulness for the scalability of attacks had not yet been suggested.

Attacks on page-based ad-blockers have unique constraints, but also enable unique exploits. Indeed, as a page-based classifier produces outputs based on a single full-page input, perturbing

content controlled by the attacker can also affect the classifier’s outputs on unperturbed page regions. The effectiveness of such attacks depends on the classifier. For the YOLOv3 [213] architecture, we show that publishers can perturb website content near ad `iframes` so as to fool the classifier into missing the actual ads (see Figure 3.10).

3.5.3 Algorithms for Adversarial Examples

For some of the considered classifiers, adversarial examples for each of the attack strategies C1-C4 in Table 3.1 can be constructed using existing and well-known techniques (we primarily use the Projected Gradient Descent attack of [159]). Below, we provide more details on the attack we use to target SIFT, and on the techniques we use to create robust and scalable attacks for page-based classifiers [272].

Black-box optimization attacks for non-parametric classifiers. SIFT is a non-parametric algorithm (i.e., with no learned parameters). As such, the standard approach for generating adversarial examples by maximizing the model’s training-loss function does not apply [246]. To remedy this, we first formulate a near-continuous loss function $L_{\text{SIFT}}(x + \delta, t)$ that acts as a proxy for SIFT’s similarity measure between the perturbed image $x + \delta$ and some fixed template t . The next difficulty is that this loss function is hard to differentiate automatically, so we use black-box optimization techniques [114] to maximize L_{SIFT} .

SIFT’s output is a variable-sized set of keypoints, where each keypoint is a vector $v \in \mathbb{R}^{132}$ —four positional values, and a 128-dimensional *descriptor* [156]. Let t be a template with keypoint descriptors T . To match an image x against t , SIFT computes descriptor vectors for x , denoted $\{v_1, \dots, v_m\}$. Then, for each v_i it finds the distances $d_{i,1}, d_{i,2}$ to its two nearest neighbors in T . The keypoint v_i is a match if the *ratio test* $d_{i,1}/d_{i,2} < \tau$ holds (where $\tau = 0.6$). Let $M(x, t)$ be the keypoints of x that match with t . To evade detection, we reduce the size of M by maximizing the following proxy loss:

$$L_{\text{SIFT}}(x + \delta, t) := \sum_{v_i \in M_\tau(x, t)} d_{i,1}/d_{i,2} . \quad (3.1)$$

Maximizing L increases $d_{i,1}/d_{i,2}$ for matched keypoints until they fall below the ratio test. To create false positives, we maximize an analogous loss that sums over $v_i \notin M_\tau(x, t)$ and decreases the ratio.

Scalable attacks with partial input control. When attacking page-based classifiers, we need to overcome two challenges: (1) the attacker only controls part of the page content and does not know which content other actors will add; (2) the attacks should be deployable at scale for a variety of web pages and ads. To create adversarial examples under these novel constraints, we combine universal [175] and transformation-robust [74, 143, 228] attacks.

To create universal perturbations, we collect additional website screenshots: D^{train} is a set of 200 screenshots of news websites, and D^{eval} contains the 20 screenshots collected in Section 3.4.1 (no

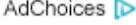
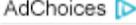
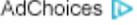
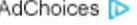
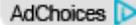
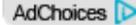
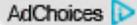
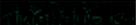
Original	Avg. Hash	OCR	SIFT
			
			
			
False Positives:			

Figure 3.6: **Adversarial examples for element-based classifiers.** These correspond to attacks (C1) and (C4) in Table 3.1.

website or ad appears in both sets). We also collect $D_{\text{BBC}}^{\text{train}}$ and $D_{\text{BBC}}^{\text{eval}}$ with 180 and 20 screenshots from `bbc.com/sports`. The training sets are used to create perturbations that work for arbitrary websites or ads. We measure attacks’ success rates on the evaluation sets.

We craft a perturbation δ by maximizing $\sum_{x \in D_*^{\text{train}}} L(x \odot \delta)$, where $x \odot \delta$ means applying the perturbation δ to a page x (note that we omit an explicit label y in the loss here, as we are not dealing with a standard classifier). Depending on the attack, the perturbation is added pixel-wise to a page region that the adversary controls, or replaces that region with δ . All that remains is the design of a suitable loss function L .

The YOLOv3 model we trained outputs multiple $B = 10,647$ boxes for detected ads, and retains a box b if its confidence—denoted $\text{conf}(f(x), b)$ —is larger than a threshold τ . To cause ads to be undetected, we thus maximize the following loss which causes all B boxes to have confidence below $\tau - \kappa$, for some slack $\kappa > 0$:

$$L_{\text{YOLO}}^{\text{FN}}(x \odot \delta) := \sum_{1 \leq b \leq B} \min((\tau - \kappa) - \text{conf}(f(x \odot \delta), b), 0), \quad (3.2)$$

For false-positives, i.e., a fake object prediction, we instead increase all boxes’ confidence up to $\tau + \kappa$ by maximizing:

$$L_{\text{YOLO}}^{\text{FP}}(x \odot \delta) := \sum_{1 \leq b \leq B} \min(\text{conf}(f(x \odot \delta), b) - (\tau + \kappa), 0). \quad (3.3)$$

3.5.4 Results

We now instantiate and evaluate the attack strategies C1-C4 from Table 3.1 on our seven ad-classifiers

Attack C1: Evasion with adversarial ad-disclosures. Figure 3.6 shows examples of perturbed AdChoices logos that fool all element-based classifiers. An ad-network can use these to evade ad-blocking.

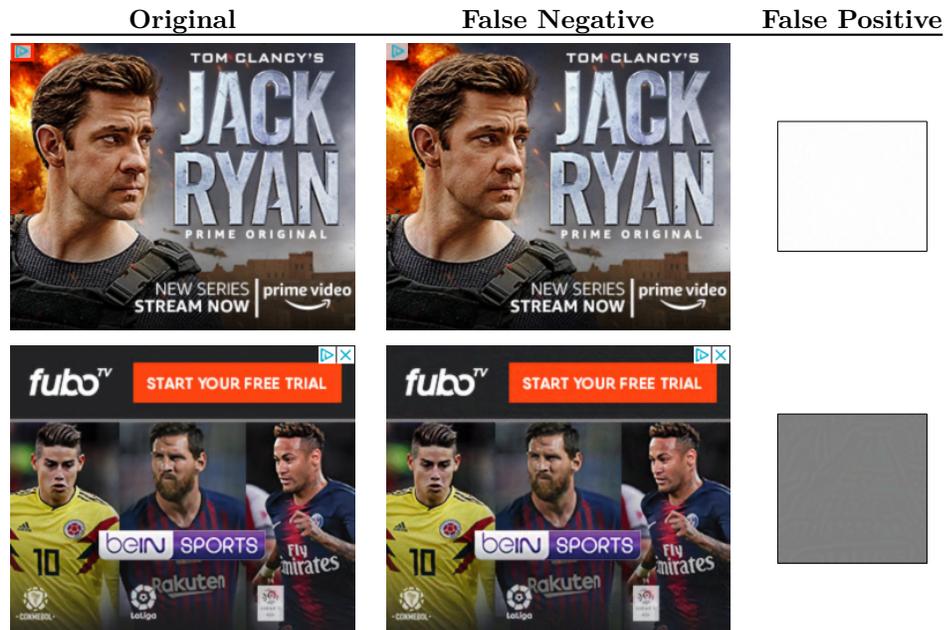


Figure 3.7: **Adversarial examples for frame-based classifiers.** These are attacks (C2) and (C4) in Table 3.1. Top: Attacks on our YOLOv3 model that detects the AdChoices logo. Bottom: attacks on the ad-classifier from [112] (we crafted similar adversarial examples for the classifier used in Percival [261])

Average hashing is invariant to small ℓ_p noise, but this comes at the cost of high sensitivity to other perturbations: we evade it by adding up to 3 transparent rows and columns to the logo. When overlaid on an ad, the rendered content is identical.

Adversarial examples for OCR bear similarities to CAPTCHAs. As ML models can solve CAPTCHAs [24, 280], one may wonder why transcribing ad disclosures is harder. The difference lies in the stronger threat model that ad-blockers face. Indeed, CAPTCHA creators have no access to the ML models they aim to fool, and must thus craft universally hard perturbations. Attacking an ad-blocker is much easier as its internal model must be public. Moreover the ad-blocker must also prevent false positives—which CAPTCHA solvers do not need to consider—and operate under stricter real-time constraints on consumer hardware.

Attack C2: Evasion with adversarial ads. Ad networks can directly perturb the ads they server to evade frame or page-based ad-blockers. For frame-based classifiers, the attacks are very simple and succeed with 100% probability (see Figure 3.7). We verified that the ad-classifier used by Percival [261] is vulnerable to similar attacks. Specifically, we create a valid HTML page containing two images—an ad and an opaque white box—which are both misclassified when the page is rendered in Percival’s modified Chromium browser (see Figure 3.8).

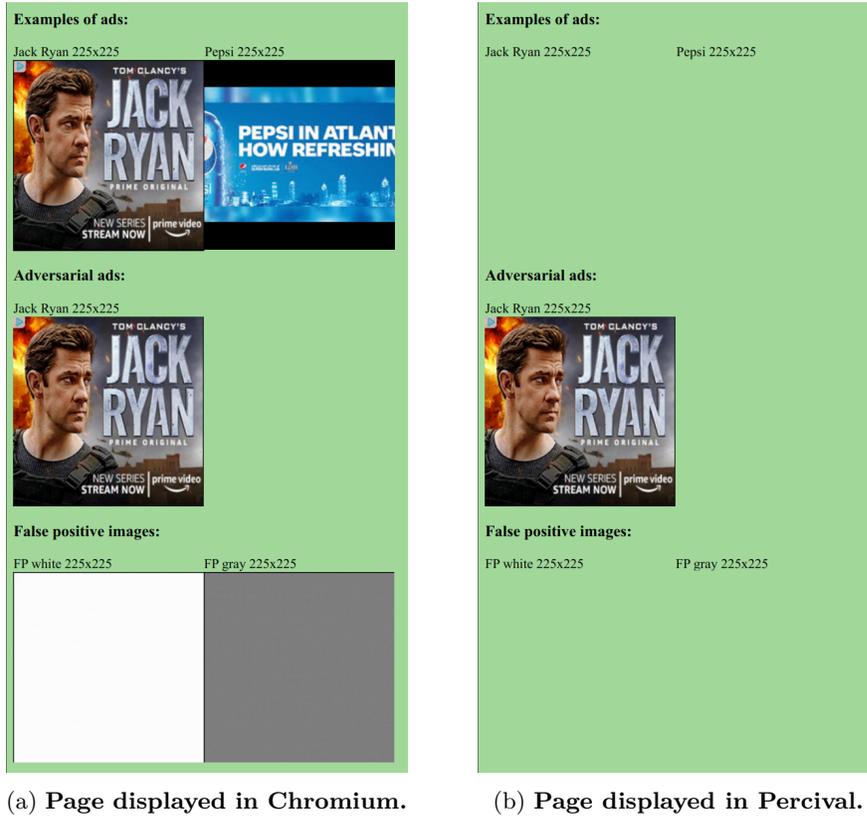


Figure 3.8: **Attack on the Percival browser from [261].** On the left, a dummy web page is displayed in the standard Chromium browser with two ads (top), an adversarially perturbed ad (middle) and two adversarial opaque boxes (bottom). On the right, the same page is displayed in the Percival browser. The two unperturbed ads on top are correctly blocked, but the adversarial ad evades detection, and the adversarial opaque boxes are mistakenly blocked.

For our page-based model, crafting a “doubly-universal” perturbation that works for all ads on all websites is hard (this is due to the model’s reliance on page layout for detecting ads, see Section 3.3 for details). Instead, we show that an ad-network can create a universal perturbation that works with 100% success rate for all ads that it serves on a specific domain (see Figure 3.10). For this attack, we maximize the $L_{\text{YOLO}}^{\text{FN}}$ loss over the collected screenshots in $D_{\text{BBC}}^{\text{train}}$, by applying the same perturbation δ over all ad frames.

Attack C3: Evasion with adversarial content. These attacks apply to page-based ad-blockers and allow publishers to evade ad-blocking while only perturbing HTML elements that they control (which crucially does not include the content of ad-frames). We show that a publisher can actually perturb the full screenshot image fed into the classifier using CSS techniques. The HTML perturbation is a *near-transparent mask*, that is overlaid on the entire web page (see Figure 3.9). The CSS

```

<div id="overlay"></div>

#overlay {
  background-image: url("data:image/png;base64,...");
  width: 100%; height: 100%; top: 0; left: 0;
  position: fixed; z-index: 10000; pointer-events: none;
  opacity: 0.01;
}

```

Figure 3.9: **Code for attack C4-U.** An adversarial mask is tiled over the full page with a small opacity factor.

properties `z-index` and `pointer-events` are used to display the mask over all other web content, but allow underlying elements to still be accessed and clicked normally.

Adding a mask over the full image is prohibitive, as the mask would be large and tied to a fixed resolution. We thus build a smaller mask and *tile* it over the full page. We generate a universal adversarial mask δ of 20KB by maximizing $L_{\text{YOLO}}^{\text{FN}}$ over D^{train} . The overlaid mask evades detection of all ads in our evaluation set (see Figure 3.11, (b)). This attack can be deployed by *any* publisher, to evade all ads. The perturbation mask is robust to scrolling and viewport changes when tested in Google Chrome.

Figure 3.11 (c) shows a similar attack that *overloads* the ad-blocker. The transparent mask is crafted to maximize $L_{\text{YOLO}}^{\text{FN}}$ over D^{train} , and creates many incorrect predictions that coerce the ad-blocker into abdicating or breaking the site. On all websites, the mask causes the model to detect abnormally large ads or fail to detect real ads.

These attacks are powerful and can be re-used by any publisher. Yet, ad-blockers might try to detect certain CSS tricks and disable them. We thus also propose stealthier attacks tuned to a single domain. For pages on BBC.com, we create a small perturbation (40×1020 px) that is applied to the white background right below an ad frame (see Figure 3.10(b)) and that universally applies to all pages from that publisher that use a similar layout.

Attack C4: Detection with adversarial honeypots To detect ad-blocking, publishers can use honeypots that falsely trigger ad-blockers [290]. The false positives in Figure 3.6 and Figure 3.7 are innocuous elements that are falsely classified as ads or ad-disclosures. For OCR and the model of Hussain et al. [112], generating near-opaque black elements worked best. As average hashing is invariant to changes in image intensity, creating false positives for it is trivial.

For page-based ad-blockers, our first attack embeds a perturbation into a small page footer (see Figure 3.11 (d)). The footer causes false predictions for 19/20 pages in our evaluation set, and is robust to a user scrolling over the page. Figure 3.10 (c) shows a stealthier attack—tailored to bbc.com—that hides a honeypot in the page header and has 100% success rate across pages from that publisher.

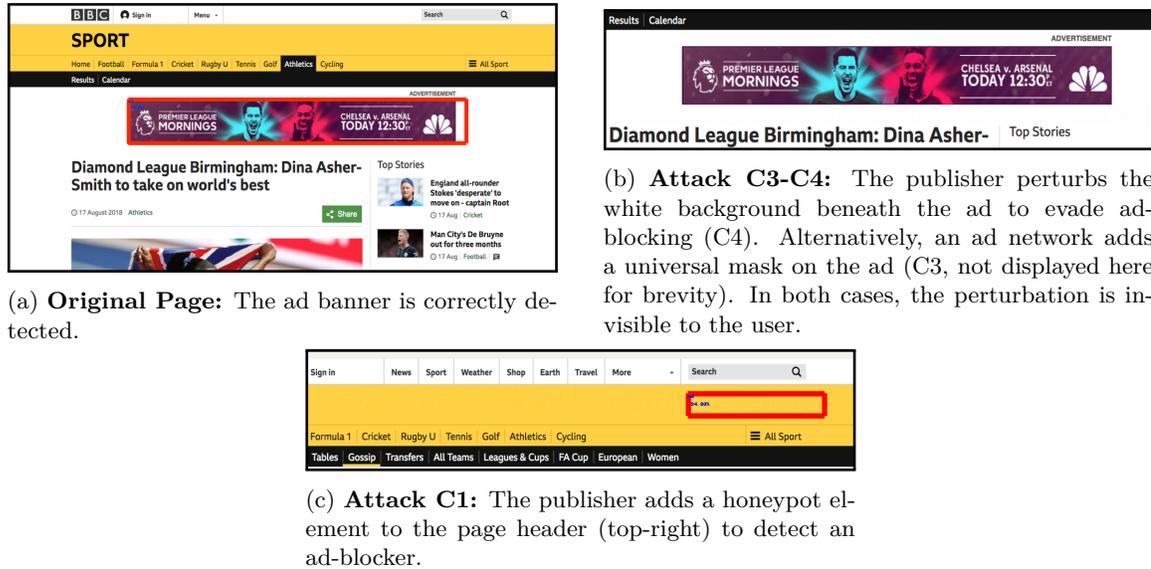


Figure 3.10: Universal adversarial examples for page-based ad-blockers on BBC.com. Examples of evasion attacks C3-C4 and detection attack C1.

3.6 Attacks Beyond Misclassification

3.6.1 Attacks Against Ad-blocker Actions

Ad-blockers usually run at a higher privilege level than any web page. They are generally not affected by the same-origin policy and can read and write any part of any web page that the user visits.

The main privileged action taken by an ad-blocker is altering of web content. Attackers exploit this action when using honeypots to detect ad-blockers. But triggering ad-blocker actions can have more pernicious effects. Below, we describe two attacks that can be deployed by *arbitrary content creators* (e.g., a Facebook user) to trigger malicious ad-blocker actions in other users' browsers.

Attack A1: Cross-boundary blocking In this attack (see Figure 3.1) a malicious user (Jerry) uploads adversarial content that triggers a Sentinel-like ad-blocker into marking content of another user (Tom) as an ad. This “cross-boundary blocking attack” hijacks the ad-blocker’s elevated privilege to bypass web security boundaries.

To mount the attack, we optimally perturb Jerry’s content so as to maximize the model’s confidence in a box that covers Tom’s content. The attack works because object-detector models such as YOLOv3 [213] predict bounding boxes by taking into account the *full* input image—a *design feature* which increases accuracy and speed [214]. As a result, adversarial content can affect bounding boxes in arbitrary image regions. Our attack reveals an inherent vulnerability of *any* object detector applied to web content—wherein the model’s segmentation misaligns with web-security boundaries.

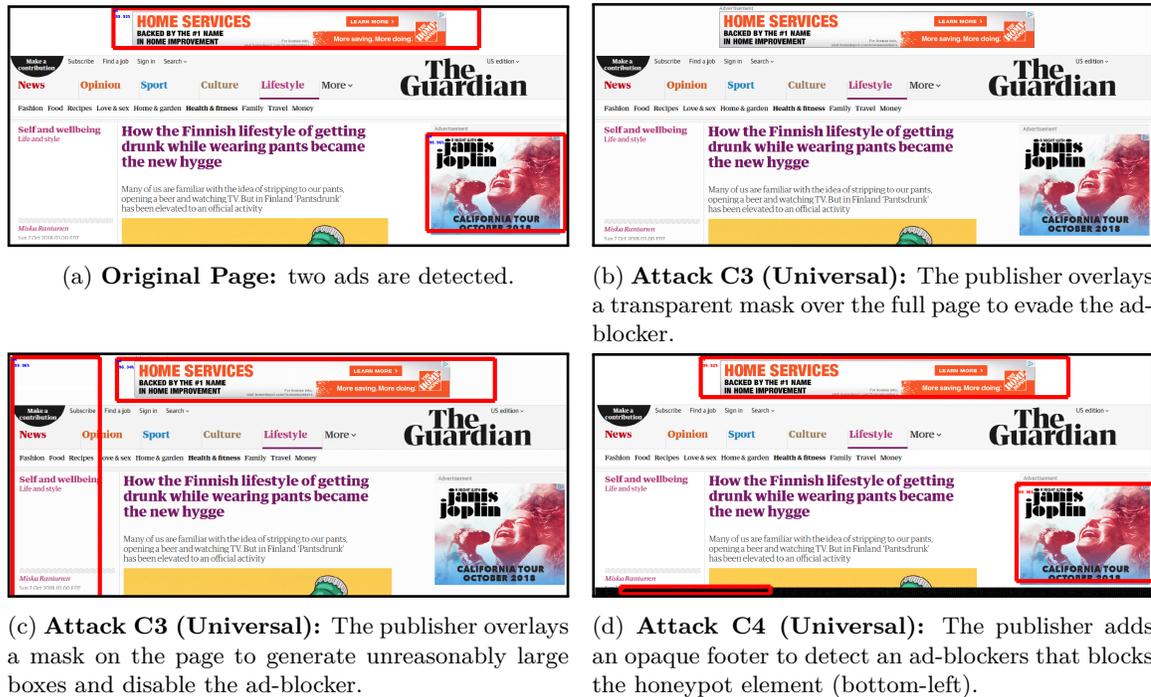


Figure 3.11: **Universal adversarial examples for page-based ad-blockers.** Displays examples of universal evasion attacks (C3) and detection attacks (C4) on a page from theguardian.com. Best viewed with 2x zoom in.

Attack A2: Cross-origin web requests In addition to searching for the “Sponsored” text on Facebook, Ad-Highlighter [243] uses the fact that the ad-disclosure contains a link to Facebook’s ad-policy page as an additional signal. Specifically, Ad-Highlighter parses the DOM in search for links containing the text “Sponsored” and determines whether the link leads to Facebook’s ad statement page by simulating a user-click on the link and following any redirects.¹⁰

These techniques are dangerous and enable serious vulnerabilities (e.g., CSRF [202], DDoS [201] or click-fraud [49]) with consequences extending beyond ad-blocking. Clicking links on a user’s behalf is a highly privileged action, which can thus be exploited by *any party that can add links in a page*, which can include arbitrary website users. To illustrate the dangers of behavioral ad-blocking, we create a regular Facebook post with an URL to a web page with title “Sponsored”. Facebook converts this URL into a link which Ad-Highlighter clicks on. *Albeit sound, this attack luckily and coincidentally fails due to Facebook’s Link Shim*, that inspects clicked links before redirecting the user. Ad-Highlighter fails to follow this particular redirection thus inadvertently preventing

¹⁰Ad-Highlighter simulates clicks because Facebook used to resolve links server-side (the ad-disclosure used to link to www.facebook.com/#). Facebook recently changed its obfuscation of the link in post captions. It now uses an empty `<a>` tag that is populated using JavaScript during the click event. This change fools Ad-Highlighter and still requires an ad-blocker to simulate a potentially dangerous click to uncover the link.

```

<a><span>
  <span class="c1">Sp</span>
  <span class="c2">S</span>
  <span class="c1">on</span>
  <span class="c2">S</span>
  <span class="c1">so</span>
  <span class="c2">S</span>
  <span class="c1">red</span>
  <span class="c2">S</span>
</span></a>

```

```

.c2 { font-size: 0; }

```



Figure 3.12: **CSS obfuscation on Facebook.com.** (Left) HTML and CSS that render Facebook’s “Sponsored” caption. (Right) A proof-of-concept where the ad-disclosure is an adversarial image that Ad-Highlighter’s OCR decodes as “8parisared”.

the attack. Yet, this also means that Facebook could use the same layer of indirection for their “Sponsored” link. If the behavioral ad-blocking idea were to be extended to disclosure cues on other websites (e.g., the AdChoices logo), such attacks would also be easily mounted. Pre-filtering inputs passed to a behavioral layer does not help. Either the filter is perfect, in which case no extra step is required—or its false positives can be exploited to trigger the behavioral component.

3.6.2 Attacks Against Page Segmentation

In this section, we describe attacks targeting the ad-blocker’s page segmentation logic, in an effort to evade the ad-blocker or exhaust its resources. These attacks use standard web techniques (e.g., HTML obfuscation) and are already applied in an ongoing arms race between Facebook and uBlock [260]. We argue that to escape the arms race caused by these segmentation attacks, perceptual ad-blockers have to operate over *rendered* web-content (i.e., frame or page-based approaches), which in turn increases the attack surface for adversarial examples on the ad-blocker’s visual classifier.

Attack S1: DOM obfuscation These attacks aim to fool the ad-blocker into feeding ambiguous inputs to its classifier. They exploit some of the same limitations that affect traditional filter lists, and can also be applied to element-based ad-blockers that rely on computer-vision classifiers, such as Ad-Highlighter.

DOM obfuscation is exemplified by Facebook’s continuous efforts to regularly alter the HTML code of its “Sponsored” caption (see Figure 3.12). Facebook deploys a variety of CSS tricks to obfuscate the caption, and simultaneously embeds hidden ad-disclosure honeypots within regular user posts in an effort to deliberately cause site-breakage for ad-block users. Facebook’s obfuscation attempts routinely fool uBlock [260] as well as Ad-Highlighter.



Figure 3.13: **Image sprites of the AdChoices logo.** *Image-sprites* are sets of images stored in a single file, and segmented using CSS rules. For example, the left sprite allows to smoothly switch from the icon to the full logo on hover. The right sprite is used by `cnn.com` to load a variety of logos used on the page in a single request.

If ad-blockers adopt computer-vision techniques as in Ad-Highlighter, DOM obfuscation attacks still apply if ad-blockers assume a direct correspondence between elements in the DOM and their visual representation when rendered. For example, Ad-Highlighter assumes that all `img` tags in the DOM are shown as is, thereby ignoring potentially complex CSS transformations applied when rendering HTML. This can cause the downstream classifier to process images with unexpected properties.

Ad networks already use CSS rules that significantly alter rendered ad-disclosures. Figure 3.13 shows two AdChoices logos found on `cnn.com`. These are image-sprites—multiple images included in a single file to minimize HTTP requests—that are cropped using CSS to display only a single logo at a time. Image-sprites highlight an exploitable blind-spot in element-based perceptual ad-blockers—e.g., the logos in Figure 3.13 fool Ad-Highlighter [243]. Images can also be fragmented into multiple elements. The ad-blocker then has to stitch them together to correctly recognize the image (e.g., Google’s AdChoices logo consists of two separate SVG tags).

Finally, the rules used by ad-blockers to link ad-disclosures back to the corresponding ad frame can also be targeted. For example, on pages with an integrated ad network, such as Facebook, the publisher could place ad-disclosures (i.e., “Sponsored” links) and ads at arbitrary places in the DOM and re-position them using CSS.

Frame-based and page-based ad-blockers bypass all these issues by operating on already-rendered content.

Attack S2: Over-segmentation Here the publisher injects a large number of elements into the DOM (say, by generating dummy images in JavaScript) to overwhelm an ad-blocker’s classifier with inputs and exhaust its resources. In response, ad-blockers would have to aggressively filter DOM elements—with the risk of these filters’ blind spots being exploited to evade or detect ad-blocking. The viability of this attack may seem unclear, as users might blame publishers for high page-load latency resulting from an overloaded ad-blocker. Yet, Facebook’s efforts to cause site-breakage by embedding ad-disclosure honeypots within all regular user posts demonstrates that some ad networks may result to such tactics.

3.6.3 Attacks Against Training

For classifiers that are trained on labeled images, the data collection and training phase can be vulnerable to *data poisoning attacks* (D1)—especially when crowdsourced as with Sentinel [272]. We describe these attacks for completeness, but refrain from a detailed evaluation as the test-time attacks described in Section 3.5 through Section 3.6.2 are conceptually more interesting and more broadly applicable.

In these attacks, the adversary joins the crowdsourced data collection to submit maliciously crafted images that adversely influence the training process. For example, malicious training data can contain *visual backdoors* [44], which are later used to evade the ad-blocker. The ad-blocker developer cannot tell if a client is contributing real data for training or malicious samples. Similar attacks against crowdsourced filter lists such as Easylist are theoretically possible. A malicious user could propose changes to filter lists that degrade their utility. However, new filters are easily interpreted and vetted before inclusion—a property not shared by visual classifiers.

Sentinel’s crowdsourced data collection of users’ Facebook feeds also raises serious privacy concerns, as a deployed model might leak parts of its training data [81, 235].

3.7 Discussion

We have presented multiple attacks to evade, detect and abuse recently proposed and deployed perceptual ad-blockers. We now provide an in-depth analysis of our results.

3.7.1 A New Arms Race

Our results indicate that perceptual ad-blocking will either perpetuate the arms race of filter lists, or replace it with an arms race around adversarial examples. Where perceptual ad-blockers that rely heavily on page markup (e.g., as in uBlock [260] or Ad-Highlighter [243]) remain vulnerable to continuous markup obfuscation [260], visual classification of rendered web content (as in Sentinel [272] or Percival [261]) inherits a crucial weakness of current visual classifiers—adversarial examples [95, 246].

The past years have seen considerable work towards mitigating the threat of adversarial examples. Yet, defenses are either broken by improved attacks [6, 26], or limited to restricted adversaries [48, 137, 159, 207, 255]. Even if ad-block developers proactively detect adversarial perturbations and blacklist them (e.g., using adversarial training [159, 246] to fine-tune their classifier), adversaries can simply regenerate new attacks (or use slightly different perturbations [230]).

3.7.2 Strategic Advantage of Adversaries and Lack of Defenses

Our attacks with adversarial examples are not a *quid pro quo* step in this new arms race, but indicate a pessimistic outcome for perceptual ad-blocking. Indeed, these ad-blockers operate in essentially *the worst threat model for visual classifiers*. Their adversaries have access to the ad-blockers’ code and prepare offline digital adversarial examples to trigger both false-negatives and false-positives in the ad-blocker’s online (and time constrained) decision making.

Even if ad-blockers obfuscate their code, black-box attacks [114] or model stealing [194, 253] still apply. Randomizing predictions or deploying multiple classifiers is also ineffective [6, 106]. For example, some of the adversarial examples in Figure 3.6 work for both OCR and SIFT despite being targeted at a single one of these classifiers.

The severity of the above threat model is apparent when considering existing defenses to adversarial examples. For instance, adversarial training [159, 246] assumes restricted adversaries (e.g., limited to ℓ_∞ perturbations), and breaks under other attacks [71, 230, 250]. Robustness to adversarial false positives (or “garbage examples” [95]) is even harder. Even if ad-blockers proactively re-train on adversarial examples deployed by publishers and ad-networks, training has a much higher cost than the attack generation and is unlikely to generalize well to new perturbations [220]. *Detecting* adversarial examples [98, 168] (also an unsolved problem [27]) is insufficient as Ad-blockers face both adversarial false-positives and false-negatives, so merely detecting an attack does not aid in decision-making. A few recently proposed defenses achieve promising results in some restricted threat models, e.g., black-box attacks [41] or physically-realizable attacks [48]. These defenses are currently inapplicable in the threat model of perceptual ad-blocking, but might ultimately reveal new insights for building more robust models.

Our attacks also apply if perceptual ad-blocking is used as a complement to filter lists rather than as a standalone approach. Ad-blockers that combine both types of techniques are vulnerable to attacks targeting either. If perceptual ad-blocking is only used passively (e.g., to aid in the maintenance of filter lists, by logging potential ads that filter lists miss), the ad-blocker’s adversaries still have incentive to attack to delay the detection of new ads.

This stringent threat model above also applies to ML-based ad-blockers that use URL and DOM features [15, 101, 119], which have not been evaluated against *adaptive white-box* attacks.

3.7.3 Beyond the Web and Vision

The use of sensory signals for ad-blocking has been considered outside the Web, e.g., AdblockRadio detects ads in radio streams using neural networks. Emerging technologies such as virtual reality [189], voice assistants [136] and smart TVs [180] are posited to become platforms for large-scale targeted advertising, and perceptual ad-blockers might emerge in those domains as well.

The threats described in this paper—and adversarial examples in particular—are likely to also affect perceptual ad-blockers that operate outside the vision domain. To illustrate, we take a closer

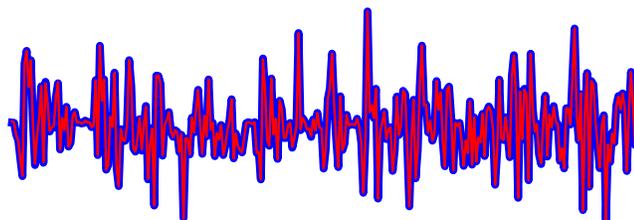


Figure 3.14: **Original and adversarial audio waveforms.** Shows a ten second segment of an ad audio waveform (thick blue) overlaid with its adversarial perturbation (thin red).

look at AdblockRadio, a radio client that continuously classifies short audio segments as speech, music or ads based on spectral characteristics. When ads are detected, the radio lowers the volume or switches stations. Radio ad-blockers face a different threat model than on the Web. All content, including ads, is served as raw audio from a single origin, so filter lists are useless. The publisher cannot run any client-side code, so ad-block detection is also impossible. Yet, the threat of adversarial examples does apply. Indeed, we show that by adding near-inaudible¹¹ noise to the ad content in AdblockRadio’s demo podcast, the perturbed audio stream evades ad detection.

Concretely, AdblockRadio takes as input a raw audio stream, computes the Mel-frequency cepstral coefficients (MFCCs), and splits them into non-overlapping windows of 4 seconds. Each segment is fed into a standard feed-forward classifier that predicts whether the segment corresponds to music, speech, or an ad. A post-processing phase merges all consecutive segments of a same class, and removes ad-segments. As the whole prediction pipeline is differentiable, crafting adversarial examples is straightforward: we use projected gradient descent (in the l_∞ norm) to modify the raw ad audio segments so as to minimize the classifier’s confidence in the ad class. The resulting audio stream fully bypasses AdblockRadio’s ad detection. An ad segment in the original and adversarial audio waveforms is displayed in Figure 3.14.

3.8 Related Work

The work in this chapter bridges two areas of computer security research—studies of the online ad-ecosystem and associated ad-blocking arms race, and adversarial examples for ML models.

Behavioral advertising. A 2015 study found that 22% of web users use ad-blockers, mainly due to intrusive behavior [138, 205, 237, 262]. The use of ad-disclosures—which some perceptual ad-blockers rely on—is rising. On the Alexa top 500, the fraction of ads with an AdChoices logo has grown from 10% to 60% in five years [108, 244]. Yet, less than 27% of users understand the logo’s meaning [148, 262].

¹¹The perturbed audio stream has a signal-to-noise ratio of 37 dB.

Ad-blocking. Limitations of filter lists are well-studied [161, 266, 270]. Many new ad-blocker designs (e.g., [15, 101, 119]) replace hard-coded rules with ML models trained on similar features (e.g., markup [57] or URLs [140]). Many of these works limit their security analysis to *non-adaptive* attacks. Ours is the first to rigorously evaluate ML-based ad-blockers.

Ad-block detection has spawned an arms race around anti-ad-blocking scripts [176, 177, 182]. Iqbal et al. [118] and Zhu et al. [290] detect anti-ad-blocking using code analysis and differential-testing. Storey et al. [244] build *stealthy* ad-blockers that aim to hide from client-side scripts, a challenging task in current browsers.

Adversarial examples. Our study of perceptual ad-blocking is the first application of adversarial examples in a real-world web-security context. Prior work attacked image classifiers [26, 95, 191, 246], malware [99], speech recognition [28] and others. We make use of white-box attacks on visual classifiers [26, 159], sequential models [28, 238] and object detectors [73]. We show that *black-box* attacks [114] are a generic alternative to prior attacks on SIFT [110].

Attacking page-based ad-blockers introduce novel challenges. Perturbing HTML bares similarities to discrete domain attacks, e.g., PDF malware detection [240]. The ad-blocker’s inputs can also be controlled by multiple entities, a constraint reminiscent of those that arise in *physical-world* attacks [7, 73, 74, 144, 228].

Preventing adversarial examples is an open problem. Adversarial training is a viable strategy [95, 144, 159, 255], but considers a less stringent threat model than perceptual ad-blockers.

3.9 Conclusion

We have presented a comprehensive security evaluation of perceptual ad-blocking. To understand the design space of these recently deployed systems, we have derived a unified architecture that incorporates and extends prior work. Our analysis of this architecture has revealed multiple vulnerabilities at every stage of the visual ad-classification pipeline. We have demonstrated that current visual ad-classifiers are inherently vulnerable to adversarial examples—the first application of these attacks to web-security. We have shown how to craft near-imperceptible perturbation for ads, ad-disclosures, and native content, in order to evade or detect ad-blocking with seven different classifiers. Finally, we have discovered a powerful attack on page-based ad-blockers, wherein a malicious user fools the model into blocking content supposedly protected by web-security boundaries.

Our aim was to highlight the fundamental vulnerabilities that perceptual ad-blockers inherit from existing image classifiers. As long as defenses to adversarial examples are elusive, perceptual ad-blockers will be dragged into a new arms race in which they start from a precariously disadvantaged position—given the stringent threat model that they must survive.

Chapter 4

Limitations of Defenses: Multiple Perturbation Types

In the following two chapters, we explore inherent limitations of current techniques that aim to make machine learning models robust to adversarial examples.

For concreteness, suppose we aimed to build a robust perceptual ad-blocker that can resist the type of attacks we have described in Chapter 3. These attacks add an imperceptible perturbation (of small ℓ_∞ norm) to the classifier’s inputs to cause a change in the classifier’s outputs. Thus, as a first goal we could consider the challenge of building classifiers that are robust to small perturbations (in the ℓ_∞ norm) of their inputs. This goal can be solved—at least partially—using successful defense techniques such as adversarial training [159] or certified defenses [207, 271].

Yet, solving this first goal is far from sufficient for building a classifier that is genuinely robust against a motivated attacker. Indeed, while we restricted ourselves to perturbations of small ℓ_∞ norm in Chapter 3, this was merely a convenience to easily convince ourselves that these perturbations are indeed imperceptible. In practice, an attacker could very well also craft perturbations that have a large ℓ_∞ norm, and yet are still imperceptible (e.g., by flipping a small number of pixels [191] or applying a minor translation to an input [71]). Thus, a robust classifier should be robust not *only* to perturbations that are small under one chosen norm type, but to a broad range of perceptually-small perturbations.

Unfortunately, existing defenses provide empirical (or certifiable) robustness guarantees for one perturbation type only, and typically offer no guarantees against other attacks [222, 230]. Worse, increasing robustness to one perturbation type has been found to increase vulnerability to others [71, 222]. This leads us to the central problem considered in this chapter:

Can we achieve adversarial robustness to different types of perturbations simultaneously?

Note that even though prior work has attained robustness to different perturbation types [71,

[159, 222], these results may not compose. For instance, an ensemble of two classifiers—each of which is robust to a single type of perturbation—may be robust to neither perturbation. Our aim is to study the extent to which it is possible to learn models that are *simultaneously* robust to multiple types of perturbation.

To gain intuition about this problem, we first study a simple and natural classification task, that has been used to analyze trade-offs between standard and adversarial accuracy [259], and the sample-complexity of adversarial generalization [220]. We define *Mutually Exclusive Perturbations (MEPs)* as pairs of perturbation types for which robustness to one type implies vulnerability to the other. For this task, we prove that ℓ_∞ and ℓ_1 perturbations are MEPs and that ℓ_∞ perturbations and input rotations and translations [71] are also MEPs. Moreover, for these MEP pairs, we find that robustness to either perturbation type requires fundamentally different features. The existence of such a trade-off for this simple classification task suggests that it may be prevalent in more complex statistical settings.

To complement our formal analysis, we introduce new adversarial training schemes for multiple perturbations. For each training point, these schemes build adversarial examples for all perturbation types and then train either on all examples (the “avg” strategy) or only the worst example (the “max” strategy). These two strategies respectively minimize the *average* error rate across perturbation types, or the error rate against an adversary that picks the worst perturbation type for each input.

For adversarial training to be practical, we also need efficient and strong attacks [159]. We show that Projected Gradient Descent [144, 159] is inefficient in the ℓ_1 case, and design a new attack, *Sparse ℓ_1 Descent (SLIDE)*, that is both efficient and competitive with strong optimization attacks [39].

We experiment with MNIST and CIFAR-10. MNIST is an interesting case-study, as *distinct* models from prior work attain strong robustness to all perturbations we consider [71, 159, 222], yet no *single* classifier is robust to all attacks [71, 221, 222]. For models trained on multiple ℓ_p attacks ($\ell_1, \ell_2, \ell_\infty$ for MNIST, and ℓ_1, ℓ_∞ for CIFAR-10), or on both ℓ_∞ and spatial transforms [71], we confirm a noticeable robustness trade-off. Figure 4.1 plots the test accuracy of models Adv_{\max} trained using our “max” strategy. In all cases, robustness to multiple perturbations comes at a cost—usually of 5-10% additional error—compared to models trained against each attack individually (the horizontal lines).

Robustness to ℓ_1, ℓ_2 and ℓ_∞ noise on MNIST is a striking failure case, where the robustness trade-off is compounded by *gradient-masking* [6, 194, 255]. Extending prior observations [149, 159, 222], we show that models trained against an ℓ_∞ adversary learn representations that *mask gradients* for attacks in other ℓ_p norms. When trained against first-order ℓ_1, ℓ_2 and ℓ_∞ attacks, the model learns to resist ℓ_∞ attacks while giving the illusion of robustness to ℓ_1 and ℓ_2 attacks. This model only achieves 52% accuracy when evaluated on gradient-free attacks [21, 222]. This shows that, unlike previously thought [259], adversarial training with strong first-order attacks can suffer from

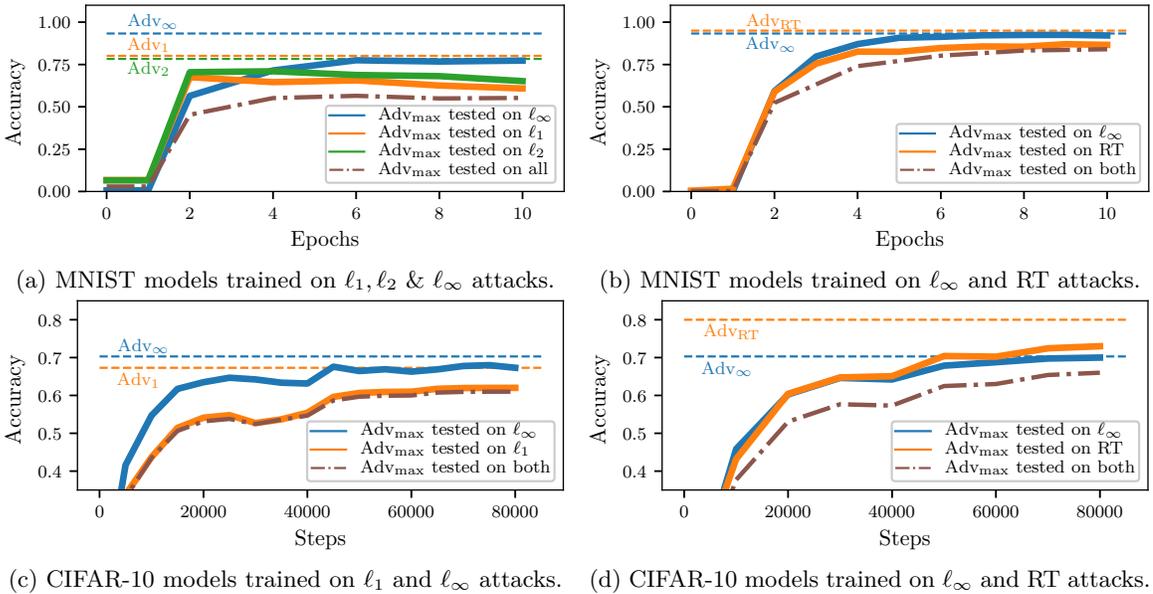


Figure 4.1: **Multi-robustness trade-off on MNIST (top) and CIFAR-10 (bottom).** For a union of ℓ_p balls (left), or of ℓ_∞ noise and rotation-translations (RT) (right), we train models Adv_{\max} on the strongest perturbation-type for each input. We report the test accuracy of Adv_{\max} against each individual perturbation type (solid line) and against their union (dotted brown line). The vertical lines show the adversarial accuracy of models trained and evaluated on a single perturbation type.

gradient-masking. We thus argue that attaining robustness to ℓ_p noise on MNIST requires new techniques (e.g., training on expensive gradient-free attacks, or scaling certified defenses to multiple perturbations).

MNIST has sometimes been said to be a poor dataset for evaluating adversarial examples defenses, as some attacks are easy to defend against (e.g., input-thresholding or binarization works well for ℓ_∞ attacks [222, 259]). Our results paint a more nuanced view: the simplicity of these ℓ_∞ defenses becomes a disadvantage when training against multiple ℓ_p norms. We thus believe that MNIST should not be abandoned as a benchmark just yet. Our inability to achieve multi- ℓ_p robustness for this simple dataset raises questions about the viability of scaling current defenses to more complex tasks.

Looking beyond adversaries that choose from a union of perturbation types, we introduce a new *affine adversary* that may linearly interpolate between perturbations (e.g., by compounding ℓ_∞ noise with a small rotation). We prove that for locally-linear models, robustness to a union of ℓ_p perturbations implies robustness to affine attacks. In contrast, affine combinations of ℓ_∞ and spatial perturbations are provably stronger than either perturbation individually. We show that this discrepancy translates to neural networks trained on real data. Thus, in some cases, attaining

robustness to a union of perturbation types remains insufficient against a more creative adversary that composes perturbations.

Our results show that despite recent successes in achieving robustness to single perturbation types, many obstacles remain towards attaining truly robust models. Beyond the robustness trade-off, efficient computational scaling of current defenses to multiple perturbations remains an open problem.

4.1 Theoretical Limits to Multi-perturbation Robustness

We study statistical properties of adversarial robustness in a natural statistical model introduced in [259], and which exhibits many phenomena observed on real data, such as trade-offs between robustness and accuracy [259] or a higher sample complexity for robust generalization [222]. This model also proves useful in analyzing and understanding adversarial robustness for multiple perturbations. Indeed, we prove a number of results that correspond to phenomena we observe on real data, in particular trade-offs in robustness to different ℓ_p or rotation-translation attacks [71].

We follow a line of works that study distributions for which adversarial examples exist *unconditionally* [75, 90, 132, 160, 225, 259]. These distributions, including ours, are much simpler than real-world data, and thus need not be evidence that adversarial examples are inevitable in practice. Rather, we hypothesize that current ML models are highly vulnerable to adversarial examples because they learn superficial data statistics [84, 115, 124] that share some properties of these simple distributions.

In prior work, a robustness trade-off for ℓ_∞ and ℓ_2 noise is shown in [132] for data distributed over two concentric spheres. Our conceptually simpler model has the advantage of yielding results beyond ℓ_p norms (e.g., for spatial attacks) and which apply symmetrically to both classes. Building on work by Xu et al. [277], Demontis et al. [60] show a robustness trade-off for dual norms (e.g., ℓ_∞ and ℓ_1 noise) in linear classifiers.

4.1.1 Adversarial Risk for Multiple Perturbation Models

We assume n *perturbation types*, each characterized by a set S of allowed perturbations for an input x . The set S can be an ℓ_p ball [95, 246] or capture other perceptually small transforms such as image rotations and translations [71]. For a perturbation $\delta \in S$, an adversarial example is $\hat{x} = x + \delta$ (this is pixel-wise addition for ℓ_p perturbations, but can be a more complex operation, e.g., for rotations).

For a perturbation set S and model f , recall the adversarial risk from Definition 2.2:

$$\mathcal{R}_{\text{adv}}(f; S) := \Pr_{(x,y) \sim \mathcal{D}} \left[\max_{\delta \in S} \mathbb{1}_{\{f(x+\delta) \neq y\}} \right]$$

To extend \mathcal{R}_{adv} to multiple perturbation sets S_1, \dots, S_n , we can consider the *average* error rate

for each S_i , denoted $\mathcal{R}_{\text{adv}}^{\text{avg}}$. This metric most clearly captures the trade-off in robustness across independent perturbation types, but is not the most appropriate from a security perspective on adversarial examples. A more natural metric, denoted $\mathcal{R}_{\text{adv}}^{\text{max}}$, is the error rate against an adversary that picks, for each input, the worst perturbation from the *union* of the S_i . More formally,

$$\mathcal{R}_{\text{adv}}^{\text{max}}(f; S_1, \dots, S_n) := \mathcal{R}_{\text{adv}}(f; \cup_i S_i), \quad \mathcal{R}_{\text{adv}}^{\text{avg}}(f; S_1, \dots, S_n) := \frac{1}{n} \sum_i \mathcal{R}_{\text{adv}}(f; S_i). \quad (4.1)$$

Most results in this section are *lower bounds* on $\mathcal{R}_{\text{adv}}^{\text{avg}}$, which also hold for $\mathcal{R}_{\text{adv}}^{\text{max}}$ since $\mathcal{R}_{\text{adv}}^{\text{max}} \geq \mathcal{R}_{\text{adv}}^{\text{avg}}$.

Two perturbation types S_1, S_2 are *Mutually Exclusive Perturbations (MEPs)*, if $\mathcal{R}_{\text{adv}}^{\text{avg}}(f; S_1, S_2) \geq 1/|C|$ for all models f (i.e., no model has non-trivial average risk against both perturbations).

4.1.2 A Binary Classification Task

We analyze the adversarial robustness trade-off for different perturbation types in a natural statistical model introduced by Tsipras et al. [259]. Their binary classification task consists of input-label pairs (x, y) sampled from a distribution \mathcal{D} as follows (note that \mathcal{D} is $(d+1)$ -dimensional):

$$y \stackrel{u.a.r.}{\sim} \{-1, +1\}, \quad x_0 = \begin{cases} +y, \text{ w.p. } p_0, \\ -y, \text{ w.p. } 1 - p_0 \end{cases}, \quad x_1, \dots, x_d \stackrel{i.i.d.}{\sim} \mathcal{N}(y\eta, 1), \quad (4.2)$$

where $p_0 \geq 0.5$, $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution and $\eta = \alpha/\sqrt{d}$ for some positive constant α .

For this distribution, Tsipras et al. [259] show a trade-off between standard and adversarial accuracy (for ℓ_∞ attacks), by drawing a distinction between the “robust” feature x_0 that small ℓ_∞ noise cannot manipulate, and the “non-robust” features x_1, \dots, x_d that can be fully overridden by small ℓ_∞ noise.

4.1.3 Small ℓ_∞ and ℓ_1 Perturbations are Mutually Exclusive

The starting point of our analysis is the observation that the robustness of a feature depends on the considered perturbation type. To illustrate, we recall two classifiers from [259] that operate on disjoint feature sets. The first, $f(x) = \text{sign}(x_0)$, achieves accuracy p_0 for all ℓ_∞ perturbations with $\varepsilon < 1$ but is highly vulnerable to ℓ_1 perturbations of size $\varepsilon \geq 1$. The second classifier, $h(x) = \text{sign}(\sum_{i=1}^d x_i)$ is robust to ℓ_1 perturbations of average norm below $\mathbb{E}[\sum_{i=1}^d x_i] = \Theta(\sqrt{d})$, yet it is fully subverted by a ℓ_∞ perturbation that shifts the features x_1, \dots, x_d by $\pm 2\eta = \Theta(1/\sqrt{d})$. We prove that this tension between ℓ_∞ and ℓ_1 robustness, and of the choice of “robust” features, is inherent for this task:

Theorem 4.1. *Let f be a classifier for \mathcal{D} . Let S_∞ be the set of ℓ_∞ bounded perturbations with $\varepsilon = 2\eta$, and S_1 the set of ℓ_1 bounded perturbations with $\varepsilon = 2$. Then, $\mathcal{R}_{\text{adv}}^{\text{avg}}(f; S_\infty, S_1) \geq 1/2$.*

The bound shows that no classifier can attain better $\mathcal{R}_{\text{adv}}^{\text{avg}}$ (and thus $\mathcal{R}_{\text{adv}}^{\text{max}}$) than a trivial constant classifier $f(x) = 1$, which satisfies $\mathcal{R}_{\text{adv}}(f; S_\infty) = \mathcal{R}_{\text{adv}}(f; S_1) = 1/2$.

Proof. Our proof follows a similar structure to the proof of Theorem 2.1 in [259], although the analysis is slightly simplified in our case as we are comparing two perturbation models, an ℓ_∞ bounded one and an ℓ_1 bounded one, that are essentially orthogonal to each other. With a perturbation of size $\varepsilon = 2\eta$, the ℓ_∞ bounded noise can “flip” the distribution of the features x_1, \dots, x_d to reflect the opposite label, and thus destroy any information that a classifier might extract from those features. On the other side, an ℓ_1 bounded perturbation with $\varepsilon = 2$ can flip the distribution of x_0 . By sacrificing some features, a classifier can thus achieve some robustness to either ℓ_∞ or ℓ_1 noise, but never to both simultaneously.

For $y \in \{-1, +1\}$, let \mathcal{G}^y be the distribution over feature x_0 conditioned on the value of y . Similarly, let \mathcal{H}^y be the conditional distribution over features x_1, \dots, x_d . Consider the following perturbations: $\delta_\infty = (0, -2y\eta, \dots, -2y\eta)$ has small ℓ_∞ norm, and $\delta_1 = (-2x_0, 0, \dots, 0)$ has small ℓ_1 norm. The ℓ_∞ perturbation can change \mathcal{H}^y to \mathcal{H}^{-y} , while the ℓ_1 perturbation can change \mathcal{G}^y to \mathcal{G}^{-y} .

Let $f(x)$ be any classifier from \mathbb{R}^{d+1} to $\{-1, +1\}$ and define:

$$p_{+-} = \Pr_{x \sim (\mathcal{G}^{+1}, \mathcal{H}^{-1})} [f(x) = +1], \quad p_{-+} = \Pr_{x \sim (\mathcal{G}^{-1}, \mathcal{H}^{+1})} [f(x) = +1].$$

The accuracy of f against the δ_∞ perturbation is given by:

$$\Pr[f(x + \delta_\infty) = y] = \Pr[y = +1] \cdot p_{+-} + \Pr[y = -1] \cdot (1 - p_{-+}) = \frac{1}{2} \cdot (1 + p_{+-} - p_{-+}).$$

Similarly, the accuracy of f against the δ_1 perturbation is:

$$\Pr[f(x + \delta_1) = y] = \Pr[y = +1] \cdot p_{-+} + \Pr[y = -1] \cdot (1 - p_{+-}) = \frac{1}{2} \cdot (1 + p_{-+} - p_{+-}).$$

Combining these, we get $\Pr[f(x + \delta_\infty) = y] + \Pr[f(x + \delta_1) = y] = 1$.

As δ_∞ and δ_1 are two specific ℓ_∞ and ℓ_1 bounded perturbations, the above is an upper-bound on the accuracy that f achieves against worst-case perturbation within the prescribed noise models, which concludes the proof. \square

Similar to [60], our analysis extends to arbitrary dual norms ℓ_p and ℓ_q with $1/p + 1/q = 1$ and $p < 2$. The perturbation required to flip the features x_1, \dots, x_n has an ℓ_p norm of $\Theta(d^{\frac{1}{p} - \frac{1}{2}}) = \omega(1)$ and an ℓ_q norm of $\Theta(d^{\frac{1}{q} - \frac{1}{2}}) = \Theta(d^{\frac{1}{2} - \frac{1}{p}}) = o(1)$. Thus, feature x_0 is more robust than features x_1, \dots, x_n with respect to the ℓ_q norm, whereas for the dual ℓ_p norm the situation is reversed.

4.1.4 Small ℓ_∞ and Spatial Perturbations are Nearly Mutually Exclusive

We now analyze two other orthogonal perturbation types, ℓ_∞ noise and rotation-translations [71]. In some cases, increasing robustness to ℓ_∞ noise has been shown to decrease robustness to rotation-translations [71]. We prove that such a trade-off is inherent for our binary classification task.

To reason about rotation-translations, we assume that the features x_i form a 2D grid. We also let x_0 be distributed as $\mathcal{N}(y, \alpha^{-2})$, a technicality that does not qualitatively change our prior results. Note that the distribution of the features x_1, \dots, x_d is permutation-invariant. Thus, the only power of a rotation-translation adversary is to “move” feature x_0 . Without loss of generality, we identify a small rotation-translation of an input x with a permutation of its features that sends x_0 to one of N fixed positions (e.g., with translations of $\pm 3\text{px}$ as in [71], x_0 can be moved to $N = 49$ different positions).

A model can be robust to these permutations by ignoring the N positions that feature x_0 can be moved to, and focusing on the remaining permutation-invariant features. Yet, this model is vulnerable to ℓ_∞ noise, as it ignores x_0 . In turn, a model that relies on feature x_0 can be robust to ℓ_∞ perturbations, but is vulnerable to a spatial perturbation that “hides” x_0 among other features. Formally, we show:

Theorem 4.2. *Let f be a classifier for \mathcal{D} (with $x_0 \sim \mathcal{N}(y, \alpha^{-2})$). Let S_∞ be the set of ℓ_∞ bounded perturbations with $\varepsilon = 2\eta$, and S_{RT} be the set of perturbations for an RT adversary with budget N . Then, $\mathcal{R}_{adv}^{avg}(f; S_\infty, S_{RT}) \geq 1/2 - O(1/\sqrt{N})$.*

Proof. The proof of this theorem follows a similar blueprint to the proof of Theorem 4.1. Recall that an ℓ_∞ perturbation with $\varepsilon = 2\eta$ can flip the distribution of the features x_1, \dots, x_n to reflect an opposite label y . The tricky part of the proof is to show that a small rotation or translation can flip the distribution of x_0 to the opposite label, without affecting the marginal distribution of the other features too much.

Recall that we model rotations and translations as picking a permutation π from some fixed set Π of permutations over the indices in x , with the constraint that feature x_0 be moved to at most N different positions for all $\pi \in \Pi$.

We again define \mathcal{G}^y as the distribution of x_0 conditioned on y , and \mathcal{H}^y for the distribution of x_1, \dots, x_d . We know that a small ℓ_∞ perturbation can transform \mathcal{H}^y into \mathcal{H}^{-y} . Our goal is to show that a rotation-translation adversary can change $(\mathcal{G}^y, \mathcal{H}^y)$ into a distribution that is very close to $(\mathcal{G}^{-y}, \mathcal{H}^y)$. The result of the theorem then follows by arguing that no binary classifier f can distinguish, with high accuracy, between ℓ_∞ perturbed examples with label y and rotated examples with label $-y$ (and vice versa).

We first describe our proof idea at a high level. We define an intermediate “hybrid” distribution \mathcal{Z}^y where all $d + 1$ features are i.i.d $N(y\eta, 1)$ (that is, x_0 now has the same distribution as the other weakly-correlated features). The main step in the proof is to show that for samples from either

$(\mathcal{G}^y, \mathcal{H}^y)$ or $(\mathcal{G}^{-y}, \mathcal{H}^y)$, a random rotation-translation yields a distribution that is very close (in total variation) to \mathcal{Z}^y . From this, we then show that there exists an adversary that applies two rotations or translations in a row, to first transform samples from $(\mathcal{G}^y, \mathcal{H}^y)$ into samples close to \mathcal{Z}^y , and then transform those samples into ones that are close to $(\mathcal{G}^{-y}, \mathcal{H}^y)$.

We will need a standard version of the Berry-Esseen theorem, stated hereafter for completeness.

Theorem 4.3 (Berry-Esseen [14]). *Let X_1, \dots, X_n be independent random variables with $\mathbb{E}[X_i] = \mu_i$, $\mathbb{E}[X_i^2] = \sigma_i^2 > 0$, and $\mathbb{E}[|X_i|^3] = \rho_i < \infty$, where the μ_i, σ_i and ρ_i are constants independent of n . Let $S_n = X_1 + \dots + X_n$, with $F_n(x)$ the CDF of S_n and $\Phi(x)$ the CDF of the standard normal distribution. Then,*

$$\sup_{x \in \mathbb{R}} \left| F_n(x) - \Phi \left(\frac{x - \mathbb{E}[S_n]}{\sqrt{\text{Var}[S_n]}} \right) \right| = O(1/\sqrt{n}).$$

For distributions \mathcal{P}, \mathcal{Q} , let $\Delta_{\text{TV}}(\mathcal{P}, \mathcal{Q})$ denote their total-variation distance. The below lemma is the main technical result we need, and bounds the total variation between a multivariate Gaussian \mathcal{P} and a special mixture of multivariate Gaussians \mathcal{Q} .

Lemma 4.4. *For $k > 1$, let \mathcal{P} be a k -dimensional Gaussians with mean $\mu_{\mathcal{P}} = (\lambda_{\mathcal{P}}, \dots, \lambda_{\mathcal{P}})$ and identity covariance. For all $i \in [k]$, let \mathcal{Q}_i be a multivariate Gaussian with mean μ_i and diagonal covariance Σ_i where*

$$(\mu_i)_j = \begin{cases} \lambda_{\mathcal{Q}} & \text{if } i = j \\ \lambda_{\mathcal{P}} & \text{otherwise} \end{cases} \quad \text{and} \quad (\Sigma_i)_{(j,j)} = \begin{cases} \sigma_{\mathcal{Q}}^2 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases}.$$

Define \mathcal{Q} as a mixture distribution of the $\mathcal{Q}_1, \dots, \mathcal{Q}_k$ with probabilities $1/k$. Assuming that $\lambda_{\mathcal{P}}, \lambda_{\mathcal{Q}}, \sigma_{\mathcal{Q}}$ are constants independent of k , we have $\Delta_{\text{TV}}(\mathcal{P}, \mathcal{Q}) = O(1/\sqrt{k})$.

*Proof of Lemma 4.4.*¹ Let $p(x)$ and $q(x)$ denote, respectively, the pdfs of \mathcal{P} and \mathcal{Q} . Note that $q(x) = \sum_{i=1}^k \frac{1}{k} q_i(x)$, where $q_i(x)$ is the pdf of \mathcal{Q}_i . We first compute:

$$\begin{aligned} q(x) &= \sum_{i=1}^k \frac{1}{k} \frac{1}{\sqrt{(2\pi)^k \cdot |\Sigma_i|}} \cdot e^{-\frac{1}{2}(x-\mu_i)^{\text{T}} \Sigma_i^{-1} (x-\mu_i)} \\ &= \frac{e^{-\frac{1}{2}(x-\mu_{\mathcal{P}})^{\text{T}} (x-\mu_{\mathcal{P}})}}{\sqrt{(2\pi)^k}} \cdot \frac{1}{k \cdot \sigma_{\mathcal{Q}}^2} \cdot \sum_{i=1}^k e^{-\frac{1}{2}t(x_i)} \\ &= p(x) \cdot \frac{1}{k \cdot \sigma_{\mathcal{Q}}^2} \cdot \sum_{i=1}^k e^{-\frac{1}{2}t(x_i)}, \end{aligned}$$

where

$$t(x_i) := (\sigma_{\mathcal{Q}}^{-2} - 1)x_i^2 - (2\lambda_{\mathcal{Q}}\sigma_{\mathcal{Q}}^{-2} - 2\lambda_{\mathcal{P}})x_i + (\lambda_{\mathcal{Q}}^2\sigma_{\mathcal{Q}}^{-2} - \lambda_{\mathcal{P}}^2). \quad (4.3)$$

¹We thank Iosif Pinelis for his help with this proof (<https://mathoverflow.net/questions/325409/>).

Thus we have that

$$q(x) < p(x) \iff \frac{1}{k \cdot \sigma_Q^2} \cdot \sum_{i=1}^k e^{-\frac{1}{2}t(x_i)} < 1.$$

The total-variation distance between \mathcal{P} and \mathcal{Q} is then $\Delta_{\text{TV}}(\mathcal{P}, \mathcal{Q}) = p_1 - p_2$, where

$$\begin{aligned} p_1 &:= \Pr [S_k < k \cdot \sigma_Q^2], \quad p_2 := \Pr [T_k < k \cdot \sigma_Q^2], \\ S_k &:= \sum_{i=1}^k U_i, \quad T_k := S_{k-1} + V_k, \quad U_i := e^{-\frac{1}{2}t(Z_i)}, \quad V_n := e^{-\frac{1}{2}t(W_n)}, \end{aligned} \tag{4.4}$$

and the $Z_i \sim \mathcal{N}(\lambda_P, 1)$, $W_n \sim \mathcal{N}(\lambda_Q, \sigma_Q^2)$ and all the Z_i and W_n are mutually independent.

It is easy to verify that $\mathbb{E}[U_i] = \sigma_Q^2$, $\text{Var}[U_i] = O(1)$, $\mathbb{E}[U_i^3] = O(1)$, $\mathbb{E}[W_n] = O(1)$, $\text{Var}[W_n] = O(1)$, $\mathbb{E}[W_n^3] = O(1)$. Then, applying the Berry-Esseen theorem, we get:

$$\begin{aligned} p_1 &= \Pr [S_k < k \cdot \sigma_Q^2] = \Phi(0) + O\left(\frac{1}{\sqrt{k}}\right) = \frac{1}{2} + O\left(\frac{1}{\sqrt{k}}\right), \\ p_2 &= \Pr [T_k < k \cdot \sigma_Q^2] = \Phi\left(\frac{k \cdot \sigma_Q^2 - \mathbb{E}[T_k]}{\sqrt{\text{Var}[T_k]}}\right) + O\left(\frac{1}{\sqrt{k}}\right) = \Phi\left(O\left(\frac{1}{\sqrt{k}}\right)\right) + O\left(\frac{1}{\sqrt{k}}\right) \\ &= \frac{1}{2} + O\left(\frac{1}{\sqrt{k}}\right). \end{aligned}$$

And thus,

$$\Delta_{\text{TV}}(\mathcal{P}, \mathcal{Q}) = p_1 - p_2 = O(1/\sqrt{k}). \tag{4.5}$$

□

We now define a rotation-translation adversary \mathcal{A} with a budget of N . It samples a random permutation from the set Π of permutations that switch position 0 with a position in $[0, N-1]$ and leave all other positions fixed (note that $|\Pi| = N$). Let $\mathcal{A}(\mathcal{G}^y, \mathcal{H}^y)$ denote the distribution resulting from applying \mathcal{A} to $(\mathcal{G}^y, \mathcal{H}^y)$ and define $\mathcal{A}(\mathcal{G}^{-y}, \mathcal{H}^y)$ similarly. Recall that \mathcal{Z}^y is a hybrid distribution which has all features distributed as $\mathcal{N}(y\eta, 1)$.

Claim 4.5. $\Delta_{\text{TV}}(\mathcal{A}(\mathcal{G}^y, \mathcal{H}^y), \mathcal{Z}^y) = O(1/\sqrt{N})$ and $\Delta_{\text{TV}}(\mathcal{A}(\mathcal{G}^{-y}, \mathcal{H}^y), \mathcal{Z}^y) = O(1/\sqrt{N})$

Proof of Claim 4.5. For the first N features, samples output by \mathcal{A} follow exactly the distribution \mathcal{Q} from Lemma 4.4, for $k = N$ and $\lambda_P = y \cdot \eta$, $\lambda_Q = y$, $\sigma_Q^2 = \alpha^{-2}$. Note that in this case, the distribution \mathcal{P} has each feature distributed as in \mathcal{Z}^y . Thus, Lemma 4.4 tells us that the distribution of the first N features is the same as in \mathcal{Z}^y , up to a total-variation distance of $O(1/\sqrt{N})$. As features $x_N \dots, x_d$ are unaffected by \mathcal{A} and thus remain distributed as in \mathcal{Z}^y , we conclude that the total-variation distance between \mathcal{A} 's outputs and \mathcal{Z}^y is $O(1/\sqrt{N})$.

The proof for $\mathcal{A}(\mathcal{G}^{-y}, \mathcal{H}^y)$ is similar, except that we apply Lemma 4.4 with $\lambda_Q = -y$. □

Let $\tilde{\mathcal{Z}}^y$ be the true distribution $\mathcal{A}(\mathcal{G}^{-y}, \mathcal{H}^y)$, which we have shown to be close to \mathcal{Z}^y . Consider the following “inverse” adversary \mathcal{A}^{-1} . This adversary samples $z \sim \tilde{\mathcal{Z}}^y$ and returns $\pi^{-1}(z)$, for $\pi \in \Pi$, with probability

$$\frac{1}{|\Pi|} \cdot \frac{f_{(\mathcal{G}^{-y}, \mathcal{H}^y)}(\pi^{-1}(z))}{f_{\tilde{\mathcal{Z}}^y}(z)},$$

where $f_{(\mathcal{G}^{-y}, \mathcal{H}^y)}$ and $f_{\tilde{\mathcal{Z}}^y}$ are the probability density functions for $(\mathcal{G}^{-y}, \mathcal{H}^y)$ and for $\tilde{\mathcal{Z}}^y$.

Claim 4.6. \mathcal{A}^{-1} is a RT adversary with budget N that transforms $\tilde{\mathcal{Z}}^y$ into $(\mathcal{G}^{-y}, \mathcal{H}^y)$.

Proof of Claim 4.6. Note that \mathcal{A}^{-1} always applies the inverse of a perturbation in Π . So feature x_0 gets sent to at most N positions when perturbed by \mathcal{A}^{-1} .

Let Z be a random variable distributed as $\tilde{\mathcal{Z}}^y$ and let h be the density function of the distribution obtained by applying \mathcal{A}^{-1} to Z . We compute:

$$\begin{aligned} h(x) &= \sum_{\pi \in \Pi} f_{\tilde{\mathcal{Z}}^y}(\pi(x)) \cdot \Pr[\mathcal{A}^{-1} \text{ picks permutation } \pi \mid Z = \pi(x)] \\ &= \sum_{\pi \in \Pi} f_{\tilde{\mathcal{Z}}^y}(\pi(x)) \cdot \frac{1}{|\Pi|} \cdot \frac{f_{(\mathcal{G}^{-y}, \mathcal{H}^y)}(\pi(\pi^{-1}(x)))}{f_{\tilde{\mathcal{Z}}^y}(\pi(x))} = \sum_{\pi \in \Pi} \frac{1}{|\Pi|} \cdot f_{(\mathcal{G}^{-y}, \mathcal{H}^y)}(x) \\ &= f_{(\mathcal{G}^{-y}, \mathcal{H}^y)}(x), \end{aligned}$$

so applying \mathcal{A}^{-1} to $\tilde{\mathcal{Z}}^y$ does yield the distribution $(\mathcal{G}^{-y}, \mathcal{H}^y)$. \square

We can now finally define our main rotation-translation adversary, \mathcal{A}^* . The adversary first applies \mathcal{A} to samples from $(\mathcal{G}^y, \mathcal{H}^y)$, and then applies \mathcal{A}^{-1} to the resulting samples from $\tilde{\mathcal{Z}}^y$.

Claim 4.7. The adversary \mathcal{A}^* is a rotation-translation adversary with budget N . Moreover,

$$\Delta_{TV}(\mathcal{A}^*(\mathcal{G}^y, \mathcal{H}^y), (\mathcal{G}^{-y}, \mathcal{H}^y)) = O(1/\sqrt{N}).$$

Proof of Claim 4.7. The adversary \mathcal{A}^* first switches x_0 with some random position in $[0, N - 1]$ by applying \mathcal{A} . Then, \mathcal{A}^{-1} either switches x_0 back into its original position or leaves it untouched. Thus, \mathcal{A}^* always moves x_0 into one of N positions. The total-variation bound follows by the triangular inequality:

$$\begin{aligned}
& \Delta_{\text{TV}}(\mathcal{A}^*(\mathcal{G}^y, \mathcal{H}^y), (\mathcal{G}^{-y}, \mathcal{H}^y)) \\
&= \Delta_{\text{TV}}(\mathcal{A}^{-1}(\mathcal{A}(\mathcal{G}^y, \mathcal{H}^y)), (\mathcal{G}^{-y}, \mathcal{H}^y)) \\
&\leq \Delta_{\text{TV}}(\mathcal{A}^{-1}(\mathcal{Z}^y), (\mathcal{G}^{-y}, \mathcal{H}^y)) + \Delta_{\text{TV}}(\mathcal{Z}^y, \mathcal{A}(\mathcal{G}^y, \mathcal{H}^y)) \\
&\leq \underbrace{\Delta_{\text{TV}}(\mathcal{A}^{-1}(\tilde{\mathcal{Z}}^y), (\mathcal{G}^{-y}, \mathcal{H}^y))}_0 + \underbrace{\Delta_{\text{TV}}(\tilde{\mathcal{Z}}^y, (\mathcal{G}^{-y}, \mathcal{H}^y))}_{O(1/\sqrt{N})} + \underbrace{\Delta_{\text{TV}}(\mathcal{Z}^y, \mathcal{A}(\mathcal{G}^y, \mathcal{H}^y))}_{O(1/\sqrt{N})} \\
&= O(1/\sqrt{N}).
\end{aligned}$$

□

To conclude the proof of Theorem 4.2, we define:

$$\begin{aligned}
p_{+-} &= \Pr_{x \sim (\mathcal{G}^{+1}, \mathcal{H}^{-1})}[f(x) = +1], & p_{-+} &= \Pr_{x \sim (\mathcal{G}^{-1}, \mathcal{H}^{+1})}[f(x) = +1], \\
\tilde{p}_{-+} &= \Pr_{x \sim \mathcal{A}^*(\mathcal{G}^{+1}, \mathcal{H}^{+1})}[f(x) = +1], & \tilde{p}_{+-} &= \Pr_{x \sim (\mathcal{G}^{-1}, \mathcal{H}^{-1})}[f(x) = +1].
\end{aligned}$$

Then,

$$\begin{aligned}
\Pr[f(x + \delta_\infty) = y] + \Pr[f(\mathcal{A}^*(x)) = y] &= \frac{1}{2}p_{+-} + \frac{1}{2}(1 - p_{-+}) + \frac{1}{2}\tilde{p}_{-+} + \frac{1}{2}(1 - \tilde{p}_{+-}) \\
&= 1 + \frac{1}{2}(p_{+-} - \tilde{p}_{+-}) + \frac{1}{2}(p_{-+} - \tilde{p}_{-+}) \\
&\leq 1 - O(1/\sqrt{N}).
\end{aligned}$$

This proof yields an asymptotic lower-bound on $\mathcal{R}_{\text{adv}}^{\text{avg}}$. We can also provide tight numerical estimates for concrete parameter settings (see [250][Appendix G.1]). □

4.1.5 Affine Combinations of Perturbations

We defined $\mathcal{R}_{\text{adv}}^{\text{max}}$ as the error rate against an adversary that may choose a different perturbation type for each input. If a model were robust to this adversary, what can we say about the robustness to a more creative adversary that *combines* different perturbation types? To answer this question, we introduce a new adversary that mixes different attacks by linearly interpolating between perturbations.

For a perturbation set S and $\beta \in [0, 1]$, we denote $\beta \cdot S$ the set of perturbations scaled down by β . For an ℓ_p ball with radius ε , this is the ball with radius $\beta \cdot \varepsilon$. For rotation-translations, the attack budget N is scaled to $\beta \cdot N$. For two sets S_1, S_2 , we define $S_{\text{affine}}(S_1, S_2)$ as the set of perturbations that compound a perturbation $\delta_1 \in \beta \cdot S_1$ with a perturbation $\delta_2 \in (1 - \beta) \cdot S_2$, for any $\beta \in [0, 1]$.

Affine combinations of ℓ_p perturbations do not affect linear models. Consider one adversary that chooses, for each input, ℓ_p or ℓ_q noise from balls S_p and S_q , for $p, q > 0$. The affine adversary picks perturbations from the set S_{affine} defined as above. We show:

Claim 4.8. *For a linear classifier $f(x) = \text{sign}(w^T x + b)$, we have $\mathcal{R}_{adv}^{max}(f; S_p, S_q) = \mathcal{R}_{adv}(f; S_{\text{affine}})$.*

Thus, for linear classifiers, robustness to a union of ℓ_p perturbations implies robustness to affine adversaries (this holds for any distribution).

Proof. Let

$$\max_{\delta \in S_U} w^T \delta = v_{\max}, \quad \text{and} \quad \min_{\delta \in S_U} w^T \delta = v_{\min} .$$

Let $S_U := S_p \cup S_q$. Note that any $\delta \in S_{\text{affine}}$ is of the form $\beta \delta_1 + (1 - \beta) \delta_2$ for $\beta \in [0, 1]$. Moreover, we have $\delta_1 \in S_p \subset S_U$ and $\delta_2 \in S_q \subset S_U$. Thus,

$$\max_{\delta \in S_{\text{affine}}} w^T \delta = v_{\max}, \quad \text{and} \quad \min_{\delta \in S_{\text{affine}}} w^T \delta = v_{\min} .$$

Let $h(x) = w^T x + b$, so that $f(x) = \text{sign}(h(x))$. Then, we get

$$\begin{aligned} \Pr_{\mathcal{D}} [\exists \delta \in S_{\text{affine}} : f(x + \delta) \neq y] &= \frac{1}{2} \Pr_{\mathcal{D}} [\exists \delta \in S_{\text{affine}} : w^T \delta < -h(x) \mid y = +1] \\ &\quad + \frac{1}{2} \Pr_{\mathcal{D}} [\exists \delta \in S_{\text{affine}} : w^T \delta > h(x) \mid y = -1] \\ &= \frac{1}{2} \Pr_{\mathcal{D}} [v_{\min} < -h(x) \mid y = +1] + \frac{1}{2} \Pr_{\mathcal{D}} [v_{\max} > h(x) \mid y = -1] \\ &= \frac{1}{2} \Pr_{\mathcal{D}} [\exists \delta \in S_U : w^T \delta < -h(x) \mid y = +1] \\ &\quad + \frac{1}{2} \Pr_{\mathcal{D}} [\exists \delta \in S_U : w^T \delta > h(x) \mid y = -1] \\ &= \Pr_{\mathcal{D}} [\exists \delta \in S_U : f(x + \delta) \neq y] . \end{aligned}$$

□

The above proof extends to models that are *locally linear* within balls S_p and S_q around the data points. For the distribution \mathcal{D} of Section 4.1.2, we can further show that there are settings (distinct from the one in Theorem 4.1) where: (1) robustness against a union of ℓ_∞ and ℓ_1 perturbations is possible; (2) this requires the model to be non-linear; (3) yet, robustness to affine adversaries is impossible (see Theorem 4.10 below). Our experiments in Section 4.3 show that neural networks trained on CIFAR-10 have a behavior that is consistent with locally-linear models, in that they are as robust to affine adversaries as against a union of ℓ_p attacks.

Affine combinations of ℓ_∞ and spatial perturbations can affect linear models. In contrast to the case above of combinations of ℓ_p and ℓ_q perturbations, compounding ℓ_∞ and spatial

perturbations yields a stronger attack, even for linear models:

Theorem 4.9. *Let $f(x) = \text{sign}(w^T x + b)$ be a linear classifier for \mathcal{D} (with $x_0 \sim \mathcal{N}(y, \alpha^{-2})$). Let S_∞ be some ℓ_∞ ball and S_{RT} be rotation-translations with budget $N > 2$. Define S_{affine} as above. Assume $w_0 > w_i > 0, \forall i \in [1, d]$. Then $\mathcal{R}_{adv}(f; S_{\text{affine}}) > \mathcal{R}_{adv}^{max}(f; S_\infty, S_{RT})$.*

This result draws a distinction between the strength of affine combinations of ℓ_p noise, and combinations of ℓ_∞ and spatial perturbations. It also shows that robustness to a union of perturbations can be insufficient against a more creative affine adversary. These results are consistent with behavior we observe in models trained on real data (see Section 4.3).

Proof. Note that our definition of affine perturbation allows for a different weighting parameter β to be chosen for each input. Thus, the adversary that selects perturbations from S_{affine} is at least as powerful as the one that selects perturbations from $S_\infty \cup S_{RT}$. All we need to show to complete the proof is that there exists some input x that the affine adversary can perturb, while the adversary limited to the union of spatial and ℓ_∞ perturbations cannot.

Without loss of generality, assume that the RT adversary picks a permutation that switches x_0 with a position in $[0, N - 1]$, and leaves all other indices untouched. The main idea is that for any input x where the RT adversary moves x_0 to position $j < N - 1$, the RT adversary with budget N is no more powerful than one with budget $j + 1$. The affine adversary can thus limit its rotation-translation budget and use the remaining budget on an extra ℓ_∞ perturbation.

We now construct an input x such that: (1) x cannot be successfully attacked by an RT adversary (with budget N) or by an ℓ_∞ adversary (with budget ε); (2) x can be attacked by an affine adversary.

Without loss of generality, assume that $w_1 = \min\{w_1, \dots, w_{N-1}\}$, i.e., among all the features that x_0 can be switched with, x_1 has the smallest weight. Let $y = +1$, and let x_1, \dots, x_{N-1} be chosen such that $\arg \min\{x_1, \dots, x_{N-1}\} = 1$. We set

$$x_0 := \frac{\varepsilon \cdot \|w\|_1}{w_0 - w_1} + x_1.$$

Moreover, set x_N, \dots, x_d such that

$$w^T x + b = 1.1 \cdot \varepsilon \cdot \|w\|_1.$$

Note that constructing such an x is always possible as we assumed $w_0 > w_i > 0$ for all $1 \leq i \leq d$.

We now have an input (x, y) that has non-zero support under \mathcal{D} . Let δ be a perturbation with $\|\delta\|_\infty \leq \varepsilon$. We have:

$$w^T(x + \delta) + b \geq w^T x + b - \varepsilon \cdot \|w\|_1 = 0.1 \cdot \varepsilon \cdot \|w\|_1 > 0,$$

so $f(w^T(x + \delta) + b) = y$, i.e., x cannot be attacked by any ε bounded ℓ_∞ perturbation.

Define \hat{x}_i as the input x with features x_0 and x_i switched, for some $0 \leq i < N$. Then,

$$\begin{aligned} w^T \hat{x}_i + b &= w^T x + b - (w_0 - w_i) \cdot (x_0 - x_i) \\ &\geq w^T x + b - (w_0 - w_1) \cdot (x_0 - x_1) \\ &= w^T x + b - \varepsilon \cdot \|w\|_1 = 0.1 \cdot \varepsilon \cdot \|w\|_1 > 0. \end{aligned}$$

Thus, the RT adversary cannot change the sign of $f(x)$ either. This means that an adversary that chooses from $S_\infty \cup S_{\text{RT}}$ cannot successfully perturb x .

Now, consider the affine adversary, with $\beta = 2/N$ that first applies an RT perturbation with budget $\frac{2}{N} \cdot N = 2$ (i.e., the adversary can only flip x_0 with x_1), followed by an ℓ_∞ perturbation with budget $(1 - \frac{2}{N}) \cdot \varepsilon$. Specifically, the adversary flips x_0 and x_1 and then adds noise $\delta = -(1 - \frac{2}{N}) \cdot \varepsilon \cdot \text{sign}(w)$. Let this adversarial example be \hat{x}_{affine} . We have

$$\begin{aligned} w^T \hat{x}_{\text{affine}} + b &= w^T x + b - (w_0 - w_1) \cdot (x_0 - x_1) - \left(1 - \frac{2}{N}\right) \cdot \varepsilon \cdot \|w\|_1 \\ &= 1.1 \cdot \varepsilon \cdot \|w\|_1 - \varepsilon \cdot \|w\|_1 - \left(1 - \frac{2}{N}\right) \cdot \varepsilon \cdot \|w\|_1 \\ &= -\left(0.9 - \frac{2}{N}\right) \cdot \varepsilon \cdot \|w\|_1 \\ &< 0. \end{aligned}$$

Thus, $f(\hat{x}_{\text{affine}}) = -1 \neq y$, so the affine adversary is strictly stronger than the adversary that is restricted to RT or ℓ_∞ perturbations. \square

Affine combinations of ℓ_p perturbations can affect non-linear models. In Claim 4.8 above, we showed that for linear models, robustness to a union of ℓ_p perturbations implies robustness to an affine adversary that interpolates between perturbation types. We show that this need not be the case when the model is non-linear. In particular, we can show that for the distribution \mathcal{D} introduced in Section 4.1, non-linearity is necessary to achieve robustness to a union of ℓ_∞ and ℓ_1 perturbations (with different parameter settings than for Theorem 4.1), but that at the same time, robustness to affine combinations of these perturbations is unattainable by any model.

Theorem 4.10. *Consider the distribution \mathcal{D} with $d \geq 200$, $\alpha = 2$ and $p_0 = 1 - \Phi(-2)$. Let S_∞ be the set of ℓ_∞ bounded perturbation with $\varepsilon = (3/2)\eta = 3/\sqrt{d}$ and let S_1 be the set of ℓ_1 bounded perturbations with $\varepsilon = 3$. Define S_{affine} as in Section 4.1.5. Then, there exists a non-linear classifier g that achieves $\mathcal{R}_{\text{adv}}^{\text{max}}(g; S_\infty, S_1) \leq 35\%$. Yet, for all classifiers f we have $\mathcal{R}_{\text{adv}}(f; S_{\text{affine}}) \geq 50\%$.*

Proof. We first prove that no classifier can achieve accuracy above 50% (which is achieved by the constant classifier) against S_{affine} . The proof is very similar to the one of Theorem 4.1.

Let $\beta = 2/3$, so the affine attacker gets to compose an ℓ_∞ budget of $2/\sqrt{d}$ and an ℓ_1 budget of 1. Specifically, for a point $(x, y) \sim \mathcal{D}$, the affine adversary will apply the perturbation

$$\delta = \left(-x_0, -y \frac{2}{\sqrt{d}}, \dots, -y \frac{2}{\sqrt{d}}\right) = (-x_0, -y\eta, \dots, -y\eta).$$

Let $\mathcal{G}^{0,0}$ be the following distribution:

$$y \stackrel{u.a.r.}{\sim} \{-1, +1\}, \quad x_0 = 0, \quad x_1, \dots, x_d \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1).$$

Note that in $\mathcal{G}^{0,0}$, x is independent of y so no classifier can achieve more than 50% accuracy on $\mathcal{G}^{0,0}$. Yet, note that the affine adversary's perturbation δ transforms any $(x, y) \sim \mathcal{D}$ into $(x, y) \sim \mathcal{G}^{0,0}$.

We now show that there exists a classifier that achieves non-trivial robustness against the set of perturbations $S_\infty \cup S_1$, i.e., the union of ℓ_∞ noise with $\varepsilon = 3/\sqrt{d}$ and ℓ_1 noise with $\varepsilon = 3$. Note that by Claim 4.8, this classifier must be *non-linear*. We define

$$f(x) = \text{sign} \left(3 \cdot \text{sign}(x_0) + \sum_{i=1}^d \frac{2}{\sqrt{d}} \cdot x_i \right).$$

The reader might notice that $f(x)$ closely resembles the *Bayes optimal classifier* for \mathcal{D} (which would be a linear classifier). The non-linearity in f comes from the sign function applied to x_0 . Intuitively, this limits the damage caused by the ℓ_1 noise, as $\text{sign}(x_0)$ cannot change by more than ± 2 under any perturbation of x_0 . This forces the ℓ_1 perturbation budget to be “wasted” on the other features x_1, \dots, x_d , which are very robust to ℓ_1 attacks.

As a warm-up, we compute the classifier's natural accuracy on \mathcal{D} . For $(x, y) \sim \mathcal{D}$, let $X = y \cdot \sum_{i=1}^d \frac{2}{\sqrt{d}} \cdot x_i$ be a random variable. Recall that $\eta = 2/\sqrt{d}$. Note that X is distributed as

$$y \cdot \sum_{i=1}^d \frac{2}{\sqrt{d}} \cdot \mathcal{N}(y\eta, 1) = \sum_{i=1}^d \frac{2}{\sqrt{d}} \cdot \mathcal{N}\left(\frac{2}{\sqrt{d}}, 1\right) = \sum_{i=1}^d \mathcal{N}\left(\frac{4}{d}, \frac{4}{d}\right) = \mathcal{N}(4, 4).$$

Recall that $x_0 = y$ with probability $p_0 = 1 - \Phi(-2) \approx 0.977$. We get:

$$\begin{aligned} \Pr_{\mathcal{D}}[f(x) = y] &= \Pr_{\mathcal{D}} \left[y \cdot \left(3 \cdot \text{sign}(x_0) + \sum_{i=1}^d \frac{2}{\sqrt{d}} \cdot x_i \right) > 0 \right] \\ &= \Pr_{\mathcal{D}}[x_0 = y] \cdot \Pr_{\mathcal{D}}[3 \cdot y \cdot \text{sign}(x_0) + X > 0 \mid x_0 = y] \\ &\quad + \Pr_{\mathcal{D}}[x_0 \neq y] \cdot \Pr_{\mathcal{D}}[3 \cdot y \cdot \text{sign}(x_0) + X > 0 \mid x_0 \neq y] \\ &= p \cdot \Pr[3 + \mathcal{N}(4, 4) > 0] + (1 - p) \cdot \Pr[-3 + \mathcal{N}(4, 4) > 0] \approx 99\%. \end{aligned}$$

We now consider an adversary that picks either an ℓ_∞ perturbation with $\varepsilon = 3/\sqrt{d}$ or an ℓ_1

perturbation with $\varepsilon = 3$. It will suffice to consider the case where $x_0 = y$. Note that the ℓ_∞ classifier cannot meaningfully perturb x_0 , and the best perturbation is always $\delta_\infty = [0, -y3/\sqrt{d}, \dots, -y3/\sqrt{d}]$. Moreover, the best ℓ_1 bounded perturbation is $\delta_1 = [-2y, -y, 0, \dots, 0]$. We have $f(x + \delta_\infty) = \text{sign}(y \cdot (3 + X - 6))$ and $f(x + \delta_1) = \text{sign}(y \cdot (-3 + X - 2/\sqrt{d}))$. We now lower-bound the classifier’s accuracy under the union $S_U := S_\infty \cup S_1$ of these two perturbation models:

$$\begin{aligned} \Pr_{\mathcal{D}}[f(x + \delta) = y, \forall \delta \in S_U] &\geq \Pr_{\mathcal{D}}[x_0 = y] \cdot \Pr_{\mathcal{D}}[f(x + \delta) = y, \forall \delta \in S_U \mid x_0 = y] \\ &\geq p \cdot \Pr_{\mathcal{D}} \left[(3 + X - 6 > 0) \wedge (-3 + X - 2/\sqrt{d} > 0) \right] \\ &= p \cdot \Pr \left[\mathcal{N}(4, 4) > 3 + 2/\sqrt{d} \right] \geq 65\% \quad (\text{for } d \geq 200). \end{aligned}$$

□

4.2 New Attacks and Adversarial Training Schemes

We complement our theoretical results with empirical evaluations of the robustness trade-off on MNIST and CIFAR-10. To this end, we first introduce new adversarial training schemes tailored to the multi-perturbation risks defined in Equation (4.1), as well as a novel attack for the ℓ_1 norm.

Multi-perturbation adversarial training. We define the empirical adversarial risk as

$$\hat{\mathcal{R}}_{\text{adv}}(f; S) = \sum_{i=1}^m \max_{\delta \in S} L(f(x^{(i)} + \delta), y^{(i)}),$$

where L is the training loss and D is the training set. For a single perturbation type, $\hat{\mathcal{R}}_{\text{adv}}$ can be minimized with *adversarial training* [159]: the maximal loss is approximated by an attack procedure $\mathcal{A}(x)$, such that $\max_{\delta \in S} L(f(x + \delta), y) \approx L(f(\mathcal{A}(x)), y)$.

For $i \in [1, d]$, let \mathcal{A}_i be an attack for the perturbation set S_i . The two multi-attack robustness metrics introduced in Equation (4.1) immediately yield the following natural adversarial training strategies:

1. **“Max” strategy:** For each input x , we train on the strongest adversarial example from all attacks, i.e., the max in $\hat{\mathcal{R}}_{\text{adv}}$ is replaced by $L(f(\mathcal{A}_{k^*}(x)), y)$, for $k^* = \arg \max_k L(f(\mathcal{A}_k(x)), y)$.
2. **“Avg” strategy:** This strategy simultaneously trains on adversarial examples from all attacks. That is, the max in $\hat{\mathcal{R}}_{\text{adv}}$ is replaced by $\frac{1}{n} \sum_{i=1}^n L(f(\mathcal{A}_i(x)), y)$.

The sparse ℓ_1 descent attack (SLIDE). Adversarial training is contingent on a *strong* and *efficient* attack. Training on weak attacks gives no robustness [255], while strong optimization

Input: Input $x \in [0, 1]^d$, steps k , step-size γ , percentile q , ℓ_1 bound ε
Output: $\hat{x} = x + \delta$ s.t. $\|\delta\|_1 \leq \varepsilon$

```

1  $\delta \leftarrow 0^d$ 
  for  $1 \leq i \leq k$  do
2    $g \leftarrow \nabla_\delta L(\theta, x + \delta, y)$ 
3    $e \leftarrow \text{sign}(g)$ 
4   for  $1 \leq j \leq d$  do  $e_j \leftarrow 0$  if  $|g_j| < P_q(\text{abs}(g))$  end
5    $\delta \leftarrow \delta + \gamma \cdot e / \|e\|_1$ 
6    $\delta \leftarrow \Pi_{S_1^\varepsilon}(\delta)$ 
  end

```

Algorithm 1: The Sparse ℓ_1 Descent Attack (SLIDE). $P_q(\text{abs}(g))$ denotes the q^{th} percentile of $\text{abs}(g)$ and $\Pi_{S_1^\varepsilon}$ is the projection onto the ℓ_1 ball (see [64]).

attacks (e.g., [26, 39]) are prohibitively expensive. Projected Gradient Descent (PGD) [144, 159] is a popular choice of attack that is both efficient and produces strong perturbations. To complement our formal results, we want to train models on ℓ_1 perturbations. Yet, we show that the ℓ_1 version of PGD is highly inefficient, and propose a better approach suitable for adversarial training.

PGD is a *steepest descent* algorithm [158]. In each iteration, the perturbation is updated in the steepest descent direction $\arg \max_{\|v\| \leq 1} v^T g$, where g is the gradient of the loss. For the ℓ_∞ norm, the steepest descent direction is $\text{sign}(g)$ [95], and for ℓ_2 , it is $g/\|g\|_2$. For the ℓ_1 norm, the steepest descent direction is the unit vector e with $e_{i^*} = \text{sign}(g_{i^*})$, for $i^* = \arg \max_i |g_i|$.

This yields an inefficient attack, as each iteration updates a single index of the perturbation δ . We thus design a new attack with finer control over the sparsity of an update step. For $q \in [0, 1]$, let $P_q(\text{abs}(g))$ be the q^{th} percentile of $\text{abs}(g)$, where $\text{abs}(g) = (|g_1|, \dots, |g_d|)$. We set $e_i = \text{sign}(g_i)$ if $|g_i| \geq P_q(\text{abs}(g))$ and 0 otherwise, and normalize e to unit ℓ_1 norm. For $q \gg 1/d$, we thus update many indices of δ at once. We introduce another optimization to handle clipping, by ignoring gradient components where the update step cannot make progress (i.e., where $x_i + \delta_i \in \{0, 1\}$ and g_i points outside the domain). To project δ onto an ℓ_1 ball, we use an algorithm of Duchi et al. [64]. Algorithm 1 describes our attack. It outperforms the steepest descent attack as well as a recently proposed Frank-Wolfe algorithm for ℓ_1 attacks [129]. Our attack is competitive with the more expensive EAD attack [39], as shown below.

Performance of the Sparse ℓ_1 Descent Attack. In Figure 4.2, we compare the performance of our new Sparse ℓ_1 Descent Attack (SLIDE) for different choices of gradient sparsity. We also compare to the standard PGD attack with the steepest-descent update rule, as well as a recent attack proposed in [129] that adapts the Frank-Wolfe optimization algorithm for finding ℓ_1 bounded adversarial examples. As we explained above, we expect our attack to outperform PGD as the steepest-descent vector is too sparse in the ℓ_1 case, and we indeed observe a significant improvement by choosing denser updates.

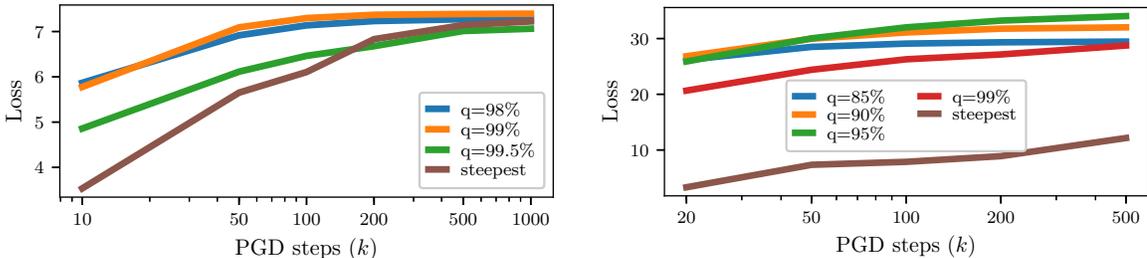


Figure 4.2: **Performance of the Sparse ℓ_1 Descent Attack on MNIST (left) and CIFAR-10 (right) for different choices of descent directions.** We run the attack for up to 1,000 steps and plot the evolution of the cross-entropy loss, for an undefended model. We vary the sparsity of the gradient updates (controlled by the parameter q), and compare to the standard PGD attack that uses the steepest descent vector, as well as the Frank-Wolfe ℓ_1 attack from [129]. For appropriate q , our attack vastly outperforms PGD and Frank-Wolfe.

The subpar performance of the Frank-Wolfe algorithm is also intriguing. We believe it is due to the attack’s linearly decreasing step-size (the k^{th} iteration has a step-size of $O(1/k)$, see [129] for details). While this choice is appropriate for optimizing convex functions, in the non-convex case it overly emphasizes the first steps of the attack, which intuitively should increase the likelihood of landing in a local minima.

4.3 Experiments

We use our new adversarial training schemes to measure the robustness trade-off on MNIST and CIFAR-10.² MNIST is an interesting case-study as *distinct* models achieve strong robustness to different ℓ_p and spatial attacks[71, 222]. Despite the dataset’s simplicity, we show that no single model achieves strong ℓ_∞, ℓ_1 and ℓ_2 robustness, and that new techniques are required to close this gap.

Training and evaluation setup. We first use adversarial training to train models on a single perturbation type. For MNIST, we use $\ell_1(\varepsilon = 10)$, $\ell_2(\varepsilon = 2)$ and $\ell_\infty(\varepsilon = 0.3)$. For CIFAR-10 we use $\ell_\infty(\varepsilon = \frac{4}{255})$ and $\ell_1(\varepsilon = \frac{2000}{255})$. We also train on rotation-translation attacks with $\pm 3\text{px}$ translations and $\pm 30^\circ$ rotations as in [71]. We denote these models $\text{Adv}_1, \text{Adv}_2, \text{Adv}_\infty$, and Adv_{RT} . We then use the “max” and “avg” strategies from Section 4.2 to train models Adv_{max} and Adv_{avg} against multiple perturbations. We train once on all ℓ_p perturbations, and once on both ℓ_∞ and RT perturbations.

²Kang et al. [129] recently studied the transfer between ℓ_∞, ℓ_1 and ℓ_2 attacks for adversarially trained models on ImageNet. They show that models trained on one type of perturbation are not robust to others, but they do not attempt to train models against multiple attacks simultaneously.

MNIST training. We use the CNN model from Madry et al. [159] and train for 10 epochs with Adam and a learning rate of 10^{-3} reduced to 10^{-4} after 5 epochs (batch size of 100). To accelerate convergence, we train against a weaker adversary in the first epoch (with 1/3 of the perturbation budget). For training, we use PGD with 40 iterations for ℓ_∞ and 100 iterations for ℓ_1 and ℓ_2 . For rotation-translations, we use the attack from [71] that picks the worst of 10 random rotation-translations.

CIFAR-10 training. We use the same wide ResNet model as [159]. We train for 80k steps of gradient descent with batch size 128 (205 epochs). When using the “avg” strategy for wide ResNet models, we had to halve the batch size to avoid overflowing the GPU’s memory. We accordingly doubled the number of training steps and learning rate schedule. We use a learning rate of 0.1 decayed by a factor 10 after 40k and 60k steps, a momentum of 0.9, and weight decay of 0.0002. Except for the RT attack, we use standard data augmentation with random padding, cropping and horizontal flipping (see [71] for details). We extract 1,000 points from the CIFAR-10 test as a validation set for early-stopping.

For training, we use PGD with 10 iterations for ℓ_∞ , and 20 iterations for ℓ_1 .³ For rotation-translations, we also use the attack from [71] that trains on the worst of 10 randomly chosen rotation-translations.

Evaluation setup. We evaluate robustness of all models using multiple attacks: (1) we use *gradient-based attacks* for all ℓ_p norms, i.e., PGD [159] and our SLIDE attack with 100 steps and 40 restarts (20 restarts on CIFAR-10), as well as Carlini and Wagner’s ℓ_2 attack [26] (C&W), and an ℓ_1 variant—EAD [39]; (2) to detect gradient-masking, we use *decision-based attacks*: the Boundary Attack [21] for ℓ_2 , the Pointwise Attack [222] for ℓ_1 , and the Boundary Attack++ [37] for ℓ_∞ ; (3) for spatial attacks, we use the optimal attack of [71] that enumerates all small rotations and translations. For unbounded attacks (C&W, EAD and decision-based attacks), we discard perturbations outside the ℓ_p ball.

For each model, we report accuracy on 1000 test points for: (1) individual perturbation types; (2) the union of these types, i.e., $1 - \mathcal{R}_{\text{adv}}^{\text{max}}$; and (3) the average of all perturbation types, $1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$. We briefly discuss the optimal error that can be achieved if there is no robustness trade-off. For perturbation sets S_1, \dots, S_n , let $\mathcal{R}_1, \dots, \mathcal{R}_n$ be the optimal risks achieved by distinct models. Then, a single model can at best achieve risk \mathcal{R}_i for each S_i , i.e., $\text{OPT}(\mathcal{R}_{\text{adv}}^{\text{avg}}) = \frac{1}{n} \sum_{i=1}^n \mathcal{R}_i$. If the errors are fully correlated, so that a maximal number of inputs admit *no* attack, we have $\text{OPT}(\mathcal{R}_{\text{adv}}^{\text{max}}) = \max\{\mathcal{R}_1, \dots, \mathcal{R}_n\}$. Our experiments show that these optimal error rates are not achieved.

³Our new attack ℓ_1 attack, described in Section 4.2, has a parameter q to controls the sparsity of the gradient updates. When leaving this parameter constant during training, the model overfits and fails to achieve general robustness. To resolve this issue, we sample $q \in [80\%, 99.5\%]$ at random for each attack during training. We also found that 10 iterations were insufficient to get a strong attack and thus increased the iteration count to 20.

Table 4.1: **Evaluation of MNIST models trained on ℓ_∞, ℓ_1 and ℓ_2 attacks (left) or ℓ_∞ and rotation-translation (RT) attacks (right).** Models $\text{Adv}_\infty, \text{Adv}_1, \text{Adv}_2$ and Adv_{RT} are trained on a single attack, while Adv_{avg} and Adv_{max} are trained on multiple attacks using the “avg” and “max” strategies. The columns show a model’s accuracy on individual perturbation types, on the union of them ($1 - \mathcal{R}_{\text{adv}}^{\text{max}}$), and the average accuracy across them ($1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$). The best results are in bold (at 95% confidence). Results in red indicate gradient-masking, see Table 4.3 for a breakdown of all attacks.

Model	Acc.	ℓ_∞	ℓ_1	ℓ_2	$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$	Model	Acc.	ℓ_∞	RT	$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$
Nat	99.4	0.0	12.4	8.5	0.0	7.0	Nat	99.4	0.0	0.0	0.0	0.0
Adv_∞	99.1	91.1	12.1	11.3	6.8	38.2	Adv_∞	99.1	91.4	0.2	0.2	45.8
Adv_1	98.9	0.0	78.5	50.6	0.0	43.0	Adv_{RT}	99.3	0.0	94.6	0.0	47.3
Adv_2	98.5	0.4	68.0	71.8	0.4	46.7	Adv_{avg}	99.2	88.2	86.4	82.9	87.3
Adv_{avg}	97.3	76.7	53.9	58.3	49.9	63.0	Adv_{max}	98.9	89.6	85.6	83.8	87.6
Adv_{max}	97.2	71.7	62.6	56.0	52.4	63.4						

4.3.1 Results on MNIST

Results are in Table 4.1. The left table is for the union of ℓ_p attacks, and the right table is for the union of ℓ_∞ and RT attacks. In both cases, the multi-perturbation training strategies “succeed”, in that models Adv_{avg} and Adv_{max} achieve higher multi-perturbation accuracy than any of the models trained against a single perturbation type.

The results for ℓ_∞ and RT attacks are promising, although the best model Adv_{max} only achieves $1 - \mathcal{R}_{\text{adv}}^{\text{max}} = 83.8\%$ and $1 - \mathcal{R}_{\text{adv}}^{\text{avg}} = 87.6\%$, which is far less than the optimal values, $1 - \text{OPT}(\mathcal{R}_{\text{adv}}^{\text{max}}) = \min\{91.4\%, 94.6\%\} = 91.4\%$ and $1 - \text{OPT}(\mathcal{R}_{\text{adv}}^{\text{avg}}) = (91.4\% + 94.6\%)/2 = 93\%$. Thus, these models do exhibit some form of the robustness trade-off analyzed in Section 4.1.

The ℓ_p results are surprisingly mediocre and re-raise questions about whether MNIST can be considered “solved” from a robustness perspective. Indeed, while training *separate* models to resist ℓ_1, ℓ_2 or ℓ_∞ attacks works well, resisting all attacks simultaneously fails. This agrees with the results of Schott et al. [222], whose models achieve either high ℓ_∞ or ℓ_2 robustness, but not both simultaneously. We show that in our case, this lack of robustness is partly due to gradient masking.

4.3.2 Results on CIFAR-10

The left table in Table 4.2 considers the union of ℓ_∞ and ℓ_1 perturbations, while the right table considers the union of ℓ_∞ and RT perturbations. As on MNIST, the models Adv_{avg} and Adv_{max} achieve better multi-perturbation robustness than any of the models trained on a single perturbation, but fail to match the optimal error rates we could hope for. For ℓ_1 and ℓ_∞ attacks, we achieve $1 - \mathcal{R}_{\text{adv}}^{\text{max}} = 61.1\%$ and $1 - \mathcal{R}_{\text{adv}}^{\text{avg}} = 64.1\%$, again significantly below the optimal values, $1 - \text{OPT}(\mathcal{R}_{\text{adv}}^{\text{max}}) = \min\{71.0\%, 66.2\%\} = 66.2\%$ and $1 - \text{OPT}(\mathcal{R}_{\text{adv}}^{\text{avg}}) = (71.0\% + 66.2\%)/2 = 68.6\%$. The results for ℓ_∞

Table 4.2: **Evaluation of CIFAR-10 models trained against ℓ_∞ and ℓ_1 attacks (left) or ℓ_∞ and rotation-translation (RT) attacks (right).** Models Adv_∞ , Adv_1 and Adv_{RT} are trained against a single attack, while Adv_{avg} and Adv_{max} are trained against two attacks using the “avg” and “max” strategies. The columns show a model’s accuracy on individual perturbation types, on the union of them ($1 - \mathcal{R}_{\text{adv}}^{\text{max}}$), and the average accuracy across them ($1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$). The best results are in bold (at 95% confidence). A breakdown of all ℓ_1 attacks is in Table 4.4.

Model	Acc.	ℓ_∞	ℓ_1	$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$	Model	Acc.	ℓ_∞	RT	$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$
Nat	95.7	0.0	0.0	0.0	0.0	Nat	95.7	0.0	5.9	0.0	3.0
Adv_∞	92.0	71.0	16.4	16.4	44.9	Adv_∞	92.0	71.0	8.9	8.7	40.0
Adv_1	90.8	53.4	66.2	53.1	60.0	Adv_{RT}	94.9	0.0	82.5	0.0	41.3
Adv_{avg}	91.1	64.1	60.8	59.4	62.5	Adv_{avg}	93.6	67.8	78.2	65.2	73.0
Adv_{max}	91.2	65.7	62.5	61.1	64.1	Adv_{max}	93.1	69.6	75.2	65.7	72.4

Table 4.3: **Breakdown of all attacks on MNIST models.** For ℓ_∞ , we use PGD and Boundary Attack++ (BAPP) [37]. For ℓ_1 , we use our Sparse ℓ_1 Descent Attack (SLIDE), EAD [39] and Pointwise Attack (PA) [222]. For ℓ_2 , we use PGD, C&W [26] and Boundary Attack (BA) [21].

Model	Acc.	ℓ_∞			ℓ_1				ℓ_2				$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$
		PGD	BAPP	All ℓ_∞	SLIDE	EAD	PA	All ℓ_1	PGD	C&W	BA	All ℓ_2		
Nat	99.4	0.0	13.0	0.0	13.0	18.8	72.1	12.4	11.0	10.4	31.0	8.5	0.0	7.0
Adv_∞	99.1	91.1	98.5	91.1	66.9	58.4	15.0	12.1	78.1	78.4	14.0	11.3	6.8	38.2
Adv_1	98.9	0.0	43.5	0.0	78.6	81.0	91.6	78.5	53.0	52.0	69.7	50.6	0.0	43.0
Adv_2	98.5	0.4	78.5	0.4	70.4	69.3	89.7	68.0	74.7	74.5	81.7	71.8	0.4	46.7
Adv_{avg}	97.3	76.7	98.0	76.7	66.3	62.4	68.6	53.9	77.7	72.3	64.6	58.3	49.9	63.0
Adv_{max}	97.2	71.7	98.5	71.7	72.1	70.0	69.6	62.6	75.7	71.8	59.7	56.0	52.4	63.4

and RT attacks are qualitatively and quantitatively similar.⁴

Models Adv_{avg} and Adv_{max} achieve 100% *training accuracy*. Thus, multi-perturbation robustness increases the *adversarial generalization gap* [220]. These models might be resorting to more memorization because they fail to find features robust to both attacks.

Detailed results for each attack. Table 4.3 and Table 4.4 below give a more detailed breakdown of each model’s accuracy against each ℓ_p attack we considered. For each model and attack, we evaluate the attack on 1,000 test points and report the accuracy. For each individual perturbation type (i.e., $\ell_\infty, \ell_1, \ell_2$), we further report the accuracy obtained by choosing the worst attack for each input. Finally, we report the accuracy against the union of all attacks ($1 - \mathcal{R}_{\text{adv}}^{\text{max}}$) as well as the average accuracy across perturbation types ($1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$).

⁴An interesting open question is why the model Adv_{avg} trained on ℓ_∞ and RT attacks does not attain optimal average robustness $\mathcal{R}_{\text{adv}}^{\text{avg}}$. Indeed, on CIFAR-10, detecting the RT attack of [71] is easy, due to the black in-painted pixels in a transformed image. The following “ensemble” model thus achieves optimal $\mathcal{R}_{\text{adv}}^{\text{avg}}$ (but not necessarily optimal $\mathcal{R}_{\text{adv}}^{\text{max}}$): on input \hat{x} , return $\text{Adv}_{\text{RT}}(\hat{x})$ if there are black in-painted pixels, otherwise return $\text{Adv}_\infty(\hat{x})$. The fact that model Adv_{avg} did not learn such a function might hint at some limitation of adversarial training.

Table 4.4: **Breakdown of all attacks on CIFAR-10 models.** For ℓ_∞ , we use PGD. For ℓ_1 , we use our Sparse ℓ_1 descent attack (SLIDE), EAD [39] and Pointwise Attack (PA) [222].

Model	Acc.	ℓ_∞		ℓ_1			All ℓ_1	$1 - \mathcal{R}_{\text{adv}}^{\text{max}}$	$1 - \mathcal{R}_{\text{adv}}^{\text{avg}}$
		PGD	All ℓ_∞	SLIDE	EAD	PA			
Nat	95.7	0.0	0.0	0.2	0.0	29.6	0.0	0.0	0.0
Adv $_\infty$	92.0	71.0	71.0	19.4	17.6	52.7	16.4	16.4	44.9
Adv $_1$	90.8	53.4	53.4	66.6	66.6	84.7	66.2	53.1	60.0
Adv $_{\text{avg}}$	91.1	64.1	64.1	61.1	61.5	81.7	60.8	59.4	62.5
Adv $_{\text{max}}$	91.2	65.7	65.7	63.1	63.0	83.4	62.5	61.1	64.1

4.3.3 First-order Adversarial Training and Gradient Masking on MNIST

On MNIST, the model Adv $_\infty$ is not robust to ℓ_1 and ℓ_2 attacks. This is unsurprising as the model was only trained on ℓ_∞ attacks. Yet, comparing the model’s accuracy against multiple types of ℓ_1 and ℓ_2 attacks (see Table 4.3) reveals a more curious phenomenon: Adv $_\infty$ has high accuracy against *first-order* ℓ_1 and ℓ_2 attacks such as PGD, but is broken by decision-free attacks. This is an indication of gradient-masking [6, 194, 255].

This issue had been observed before [149, 222], but an explanation remained illusive, especially since ℓ_∞ PGD does not appear to suffer from gradient masking (see [159]). We explain this phenomenon by inspecting the learned features of model Adv $_\infty$, as in [159]. We find that the model’s first layer learns threshold filters $\mathbf{z} = \text{ReLU}(\alpha \cdot (x - \varepsilon))$ for $\alpha > 0$. As most pixels in MNIST are zero, most of the z_i cannot be activated by an ε bounded ℓ_∞ attack. The ℓ_∞ PGD thus optimizes a smooth (albeit flat) loss function. In contrast, ℓ_1 and ℓ_2 attacks can move a pixel $x_i = 0$ to $\hat{x}_i > \varepsilon$ thus activating z_i , but have no gradients to rely on (i.e. $dz_i/dx_i = 0$ for any $x_i \leq \varepsilon$). Figure 4.3 shows that the model’s loss resembles a step-function, for which first-order attacks such as PGD are inadequate.

Note that training against first-order ℓ_1 or ℓ_2 attacks directly (i.e., models Adv $_1$ and Adv $_2$ in Table 4.1), seems to yield genuine robustness to these perturbations. This is surprising in that, because of gradient masking, model Adv $_\infty$ actually achieves lower training loss against first-order ℓ_1 and ℓ_2 attacks than models Adv $_1$ and Adv $_2$. That is, Adv $_1$ and Adv $_2$ converged to sub-optimal local minima of their respective training objectives, yet these minima generalize much better to stronger attacks.

The models Adv $_{\text{avg}}$ and Adv $_{\text{max}}$ that are trained against ℓ_∞, ℓ_1 and ℓ_2 attacks also learn to use thresholding to resist ℓ_∞ attacks while spuriously masking gradient for ℓ_1 and ℓ_2 attacks. This is evidence that, unlike previously thought [259], training against a strong first-order attack (such as PGD) can cause the model to minimize its training loss via gradient masking. To circumvent this issue, alternatives to first-order adversarial training seem necessary. Potential (costly) approaches include training on gradient-free attacks, or extending certified defenses [207, 271] to multiple perturbations. Certified defenses provide provable bounds that are much weaker than the robustness

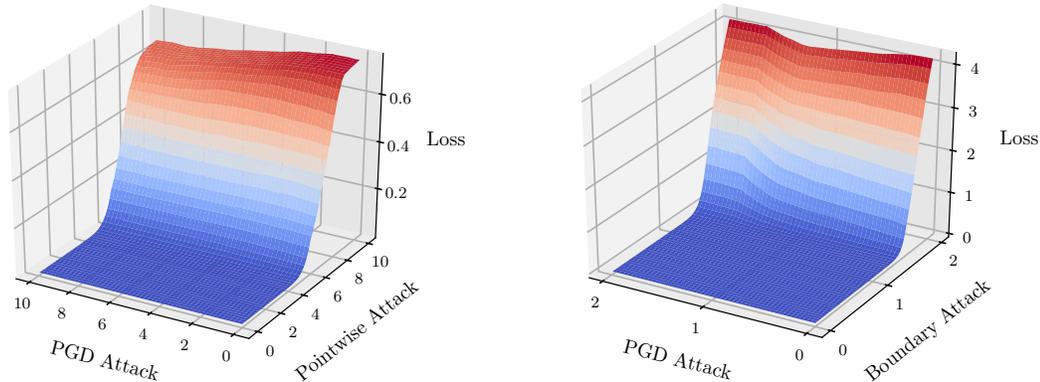


Figure 4.3: **Gradient masking in an ℓ_∞ adversarially trained model on MNIST, evaluated against ℓ_1 attacks (left) and ℓ_2 attacks (right).** The model is trained against an ℓ_∞ PGD adversary with $\varepsilon = 0.3$. For a randomly chosen data point x , we compute an adversarial perturbation δ_{PGD} using PGD and δ_{GF} using a gradient-free attack. The left plot is for ℓ_1 attacks with $\varepsilon = 10$ and the right plot is for ℓ_2 attacks with $\varepsilon = 2$. The plots display the loss on points of the form $\hat{x} := x + \alpha \cdot \delta_{\text{PGD}} + \beta \cdot \delta_{\text{GF}}$, for $\alpha, \beta \in [0, \varepsilon]$. The loss surface behaves like a step-function, and gradient-free attacks succeed in finding adversarial examples where first-order methods failed.

Table 4.5: **Evaluation of affine attacks.** For models trained with the “max” strategy, we evaluate against attacks from a union S_U of perturbation sets, and against an affine adversary that interpolates between perturbations. Examples of affine attacks are in Figure 4.4.

Dataset	Attacks	acc. on S_U	acc. on S_{affine}
MNIST	ℓ_∞ & RT	83.8	62.6
CIFAR-10	ℓ_∞ & RT	65.7	56.0
CIFAR-10	ℓ_∞ & ℓ_1	61.1	58.0

attained by adversarial training, and certifying multiple perturbation types is likely to exacerbate this gap.

4.3.4 Affine Adversaries

Finally, we evaluate the affine attacks introduced in Section 4.1.5. These attacks take affine combinations of two perturbation types, and we apply them on the models Adv_{max} (we omit the ℓ_p case on MNIST due to gradient masking). To compound ℓ_∞ and ℓ_1 noise, we devise an attack that updates both perturbations in alternation. To compound ℓ_∞ and RT attacks, we pick random rotation-translations (with $\pm 3\beta$ px translations and $\pm 30\beta^\circ$ rotations), apply an ℓ_∞ attack with budget $(1 - \beta)\varepsilon$ to each, and retain the worst example. In Figure 4.4, we display examples of ℓ_1 , ℓ_∞ and rotation-translation attacks on MNIST and CIFAR-10, as well as affine attacks that interpolate between two attack types.

The results in Table 4.5 match the predictions of our formal analysis: (1) affine combinations

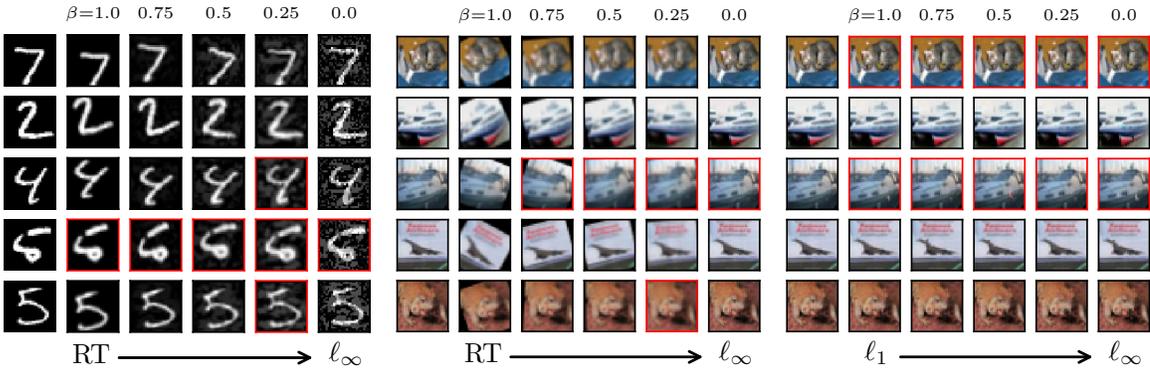


Figure 4.4: **Adversarial examples for ℓ_∞ , ℓ_1 and rotation-translation (RT) attacks, and affine combinations thereof.** The first column in each subplot shows clean images. The following five images in each row linearly interpolate between two attack types, as described in Section 4.1.5. Images marked in red are mis-classified by a model trained against both types of perturbations. Note that there are examples for which combining a rotation-translation and ℓ_∞ attack is stronger than either perturbation type individually.

of ℓ_p perturbations are no stronger than their union. This is expected given Claim 4.8 and prior observations that neural networks are close to linear near the data [95, 215]; (2) combining of ℓ_∞ and RT attacks does yield a stronger attack, as shown in Theorem 4.9. This demonstrates that robustness to a union of perturbations can still be insufficient to protect against more complex combinations of perturbations.

4.4 Discussion and Open Problems

Despite recent success in defending ML models against some perturbation types [71, 159, 222], extending these defenses to multiple perturbations unveils a clear robustness trade-off. This tension may be rooted in its unconditional occurrence in natural and simple distributions, as we proved in Section 4.1.

Our new adversarial training strategies fail to achieve competitive robustness to more than one attack type, but narrow the gap towards multi-perturbation robustness. We note that the optimal risks \mathcal{R}_{adv}^{max} and \mathcal{R}_{adv}^{avg} that we achieve are very close. Thus, for most data points, the models are either robust to all perturbation types or none of them. This hints that some points (sometimes referred to as *prototypical examples* [30, 241]) are inherently easier to classify robustly, regardless of the perturbation type.

We showed that first-order adversarial training for multiple ℓ_p attacks suffers from gradient masking on MNIST. Achieving better robustness on this simple dataset is an open problem. Another challenge is reducing the cost of our adversarial training strategies, which scale linearly in the number of perturbation types. Breaking this linear dependency requires efficient techniques for finding

perturbations in a union of sets, which might be hard for sets with near-empty intersection (e.g., ℓ_∞ and ℓ_1 balls). The cost of adversarial training has also been reduced by merging the inner loop of a PGD attack and gradient updates of the model parameters [226, 285], but it is unclear how to extend this approach to a union of perturbations (some of which are not optimized using PGD, e.g., rotation-translations).

Hendrycks and Dietterich [107], and Geirhos et al. [84] recently measured robustness of classifiers to multiple common (i.e., non-adversarial) image corruptions (e.g., random image blurring). In that setting, they also find that different classifiers achieve better robustness to some corruptions, and that no single classifier achieves the highest accuracy under all forms. The interplay between multi-perturbation robustness in the adversarial and common corruption case is worth further exploration.

Chapter 5

Limitations of Defenses: Excessive Invariance

The work we presented in Chapter 4 shows that building a classifier that is robust against multiple types of small perturbations is a remarkable challenge. As a result, all current defenses remain inherently vulnerable to simple attacks.

In this chapter, we argue that this problem is not merely technical. That is, even if we did manage to train models that are robust against all perturbations from, say, a union of ℓ_p balls, we would still face major challenges towards achieving meaningful robustness. To motivate the discussion in this chapter, consider the following seemingly benign, yet critically important question:

How large of a perturbation set should our models be made robust to?

Answering this question is challenging, since the perturbation types we have considered (e.g., small ℓ_p balls, or small rotations and translations) are only a crude approximation to the true visual similarity in a given task. In this chapter, we show that optimizing a model’s robustness to such perturbations is not only insufficient to resist general adversarial examples, but also potentially *harmful*. If a model attains robustness to large enough perturbations, we find that it may become excessively invariant to real semantics of the underlying task.

Excessive invariance of a model causes vulnerability to *invariance adversarial examples* [120]. These are perturbations that change the human-assigned label of an input but keep the model prediction unchanged. For example, in Figure 5.1 an image of a digit ‘3’ is perturbed to be an image of a ‘5’ by changing only 20 pixels; models that are excessively invariant do not change their decision and incorrectly label both images as a ‘3’, despite the fact that the oracle label has changed.

In this chapter, we will distinguish such invariance-based adversarial examples from the traditional *sensitivity-based* adversarial examples we have considered so far (where a small perturbation causes a change in the model’s output).

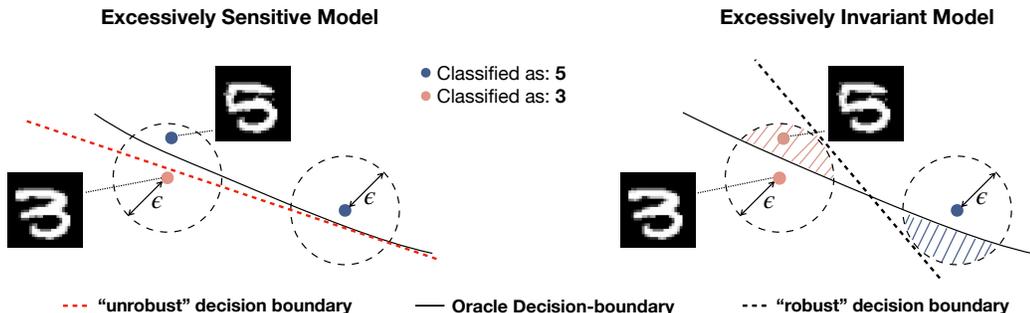


Figure 5.1: **Decision boundaries near a real image of a digit ‘3’ and an invariance-based adversarial example labeled as ‘5’.** [Left]: Training a classifier without constraints may learn a decision boundary unrobust to sensitivity-based adversarial examples. [Right]: Enforcing robustness to norm-bounded perturbations introduces erroneous invariance (dashed regions in ϵ -spheres). We display real data here, the misclassified ‘5’ is an image found by our attack which resides within a typically reported ϵ -region around the displayed ‘3’ (in the ℓ_0 norm). This excessive invariance of the robust model in task-relevant directions illustrates how robustness to sensitivity-based adversarial examples can result in new model vulnerabilities.

We expose a fundamental tradeoff between sensitivity-based and invariance-based adversarial examples. We show that due to a misalignment between formal robustness notions (e.g., ℓ_p balls) and a task’s perceptual metric, current defenses against adversarial examples cannot prevent both sensitivity-based and invariance-based attacks, and must trade-off robustness to each (see Figure 5.2). Worse, we find that increasing robustness to sensitivity-based attacks **decreases** a model’s robustness to invariance-based attacks. We introduce new algorithms to craft ℓ_p bounded invariance-based adversarial examples, and illustrate the above tradeoff on MNIST.¹ We show that state-of-the-art robust models disagree with human labelers on many of our crafted invariance-based examples, and that the disagreement rate is higher the more robust a model is. We find that even models robust to very small perturbations (e.g., of ℓ_∞ norm below $\epsilon = 0.1$) have higher vulnerability to invariance attacks compared to undefended models.

We further break a *provably-robust* defense [286] with our attack. This model is certified to have 87% test-accuracy (with respect to the MNIST test-labels) under ℓ_∞ noise of radius $\epsilon = 0.4$. That is, for 87% of test inputs (x, y) , the model is guaranteed to predict class y for any perturbed input \hat{x} that satisfies $\|x - \hat{x}\|_\infty \leq 0.4$. Yet, on our invariance-based adversarial examples that satisfy this norm-bound, the model only agrees with human labelers in 60% of the cases for an automated attack, and 12% of the cases for manually-created examples—i.e., no better than chance. The reason is that we can find perturbed inputs \hat{x} that humans no longer classify the same way as x .

Finally, we introduce a classification task where the tradeoff between sensitivity and invariance can be studied rigorously. We show that excessive sensitivity and invariance are tied respectively

¹While MNIST can be a poor choice for studying adversarial examples, we chose it because it is the *only* vision task for which models have been made robust in non-negligible ℓ_p norm balls. The fundamental tradeoff described in this chapter will affect other vision tasks once we can train strongly robust models on them.

to the existence of generalizable non-robust features [115, 124, 281] and to *robust features* that are predictive for standardized benchmarks, but not for the general vision tasks that these benchmarks aim to capture. Our experiments on MNIST show that such *overly-robust* features exist. We further argue formally and empirically that data augmentation may offer a solution to both excessive sensitivity and invariance.

5.1 Norm-bounded Sensitivity and Invariance Attacks

We begin by defining a framework to formally describe two complementary failure modes of machine learning models, namely (norm-bounded) adversarial examples that arise from excessive *sensitivity* or *invariance* of a classifier.

We extend the definition of an adversarial example given in Definition 2.1 by incorporating an explicit *labeling oracle* $\mathcal{O} : \mathbb{R}^d \rightarrow [C] \cup \{\perp\}$ that maps any input in \mathbb{R}^d to its true label, or to the “garbage class” \perp for inputs x considered “un-labelable” (e.g., for a digit classification task, the oracle \mathcal{O} corresponds to human-labeling of any image as a digit or as the garbage class). Note that for $(x, y) \sim \mathcal{D}$, we always have $y = \mathcal{O}(x)$.²

The goal of robust classification is then to learn a classifier $f : \mathbb{R}^d \rightarrow [C]$ that agrees with the oracle’s labels not only in expectation over the distribution \mathcal{D} , but also on any rare or out-of-distribution inputs to which the oracle assigns a class label—including adversarial examples obtained by imperceptibly perturbing inputs sampled from \mathcal{D} .

At its broadest, the definition of an adversarial example encompasses any adversarially induced failure in a classifier [94]. That is, an adversarial example is any input \hat{x} created such that $f(\hat{x}) \neq \mathcal{O}(\hat{x})$. This definition has proven difficult to work with, due to its inherent reliance on the oracle \mathcal{O} . As a result, it has become customary to study a relaxation of this definition, which restricts the adversary to applying a “small” perturbation to an input x sampled from the distribution \mathcal{D} . A common choice is to restrict the adversary to perturbations from some set S , e.g., a small ℓ_p ball. This recovers the definition of an adversarial example we have used so far, which we will call “sensitivity adversarial examples”:

Definition 5.1 (Sensitivity Adversarial Examples). Given a classifier f and a correctly classified input $(x, y) \sim \mathcal{D}$ (i.e., $\mathcal{O}(x) = f(x) = y$), an ε -bounded sensitivity adversarial example is an input $\hat{x} \in \mathbb{R}^d$ such that:

1. $f(\hat{x}) \neq f(x)$.
2. $\|\hat{x} - x\| \leq \varepsilon$.

²We view the support of \mathcal{D} as a strict subset of all inputs in \mathbb{R}^d to which the oracle assigns a label. That is, there are inputs for which humans agree on a label, yet that have measure zero in the data distribution from which the classifier’s train and test inputs are chosen. For example, the train-test data is often a sanitized and normalized subset of natural inputs. Moreover, “unnatural” inputs such as adversarial examples might never arise in natural data.

The assumption underlying this definition is that perturbations satisfying $\|\hat{x} - x\| \leq \varepsilon$ preserve the oracle’s labeling of the original input x , i.e., $\mathcal{O}(\hat{x}) = \mathcal{O}(x)$. If this assumption holds, then Definition 5.1 is equivalent to Definition 2.1 that we have considered thus far. In this chapter, we will primarily be interested in cases where this assumption might get violated.

A long line of work studies techniques to make classifiers robust to norm-bounded sensitivity adversarial examples [95, 159]. The main objective of these works is to minimize a classifier’s *adversarial risk* under ε -bounded perturbations:

$$\mathcal{R}_{\text{adv}}(f; \varepsilon) := \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\max_{\|\delta\| \leq \varepsilon} \mathbb{1}_{\{f(x+\delta) \neq y\}} \right]. \quad (5.1)$$

Note that $\mathcal{R}_{\text{adv}}(f; \varepsilon)$ is syntactic sugar for the general form of adversarial risk in Definition 2.2 with the perturbation set $S = \{\delta : \|\delta\| \leq \varepsilon\}$.

We study a complementary failure mode to sensitivity adversarial examples, called invariance adversarial examples [120]. These correspond to (bounded) perturbations that *do not* preserve an input’s oracle-assigned label, yet preserve the model’s classification:

Definition 5.2 (Invariance Adversarial Examples). Given a classifier f and a correctly classified input $(x, y) \sim \mathcal{D}$, an ε -bounded invariance adversarial example is an input $\hat{x} \in \mathbb{R}^d$ such that:

1. $f(\hat{x}) = f(x)$.
2. $\mathcal{O}(\hat{x}) \neq \mathcal{O}(x)$, and $\mathcal{O}(\hat{x}) \neq \perp$.
3. $\|\hat{x} - x\| \leq \varepsilon$.

If the assumption on sensitivity adversarial examples in Definition 5.1 is met—i.e., all ε -bounded perturbations preserve the label—then Definition 5.1 and Definition 5.2 correspond to well-separated failure modes of a classifier (i.e., ε' -bounded invariance adversarial examples only exist for $\varepsilon' > \varepsilon$).

Our main contribution is to reveal fundamental trade-offs between these two types of adversarial examples, that arise from this assumption being violated. We demonstrate that state-of-the-art robust classifiers do violate this assumption, and (sometimes certifiably) have low robust error $\mathcal{R}_{\text{adv}}(f; \varepsilon)$ for a norm-bound ε that does not guarantee that the oracle’s label is preserved. We show that these classifiers actually have high “true” robust error as measured by human labelers.

Remarks. Definition 5.2 is a conscious restriction on a definition of Jacobsen et al. [120], who define an invariance adversarial example as an *unbounded* perturbation that changes the oracle’s label while preserving a classifier’s output at *an intermediate feature layer*. As we solely consider the model’s final classification, considering unbounded perturbations would allow for a “trivial” attack: given an input x of class y , find any input of a different class that the model misclassifies as y . (e.g., given an image of a digit 8, an unbounded invariance example could be any unperturbed digit that the classifier happens to misclassify as an 8).

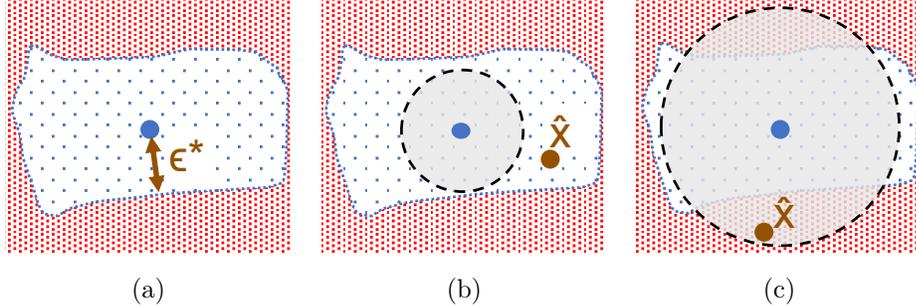


Figure 5.2: **Illustration of distance-oracle misalignment.** The input space is (ground-truth) classified into the red solid region, and the white dotted region. (a) A point at distance ε^* (under a chosen norm) of the oracle decision boundary. (b) A model robust to perturbations of norm $\varepsilon \leq \varepsilon^*$ (gray circle) is still overly sensitive and can have adversarial examples \hat{x} . (c) A model robust to perturbations of norm $\varepsilon > \varepsilon^*$ (gray circle) has invariance adversarial examples \hat{x} .

Definition 5.2 presents the same difficulty as the original broad definition of adversarial examples: a dependence on the oracle \mathcal{O} . Automating the process of finding invariance adversarial examples is thus challenging. In Section 5.3.2, we present some successful automated attacks, but show that a human-in-the-loop process is more effective.

5.2 The Sensitivity and Invariance Tradeoff

In this section, we show that if the norm that is used to define “small” adversarial perturbations is misaligned with the labeling oracle \mathcal{O} , then the robust classification objective in Equation (5.1) is insufficient for preventing both sensitivity-based and invariance-based adversarial examples under that norm. That is, we show that optimizing a model to attain low robust error on perturbations of norm ε cannot prevent both sensitivity and invariance adversarial examples.

We begin by formalizing our notion of norm-oracle misalignment. The definition applies to any similarity metric over inputs, of which ℓ_p norms are a special case.

Definition 5.3 (Distance-Oracle Misalignment). Let $\text{dist} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a distance measure (e.g., $\|x^{(1)} - x^{(2)}\|$). We say that dist is aligned with the oracle \mathcal{O} if for any input x with $\mathcal{O}(x) = y$, and any inputs $x^{(1)}, x^{(2)}$ such that $\mathcal{O}(x^{(1)}) = y$, $\mathcal{O}(x^{(2)}) \neq y$, we have $\text{dist}(x, x^{(1)}) < \text{dist}(x, x^{(2)})$. dist and \mathcal{O} are misaligned if they are not aligned.

For natural images, ℓ_p norms (or other simple metrics) are clearly misaligned with our own perceptual metric. A concrete example is in Figure 5.3. This simple fact has deep implications for the suitability of the robust classification objective in Equation (5.1). For an input $(x, y) \sim \mathcal{D}$, we define the size of the smallest class-changing perturbation as:

$$\varepsilon^*(x) := \min \{ \|\delta\| : \mathcal{O}(x + \delta) \notin \{y, \perp\} \} . \quad (5.2)$$

Let x be an input where the considered distance function is not aligned with the oracle. Let $x^{(2)} = x + \delta$ be the closest input to x with a different class label, i.e., $\mathcal{O}(x^{(2)}) = y' \neq y$ and $\|\delta\| = \varepsilon^*(x)$. As the distance and oracle are misaligned, there exists an input $x^{(1)} = x + \delta'$ such that $\|\delta'\| > \varepsilon^*(x)$ and $\mathcal{O}(x^{(1)}) = y$. So now, if we train a model to be robust (in the sense of Equation (5.1)) to perturbations of norm bounded by $\varepsilon \leq \varepsilon^*(x)$, the model might misclassify $x^{(1)}$, i.e., it is sensitive to non-semantic changes. Instead, if we make the classifier robust to perturbations bounded by $\varepsilon > \varepsilon^*(x)$, then $x^{(2)}$ becomes an invariance adversarial examples as the model will classify it the same way as x . The two types of failure modes are visualized in Figure 5.2.

Lemma 5.4. *Constructing an oracle-aligned distance function that satisfies Definition 5.3 is as hard as constructing a function f so that $f(x) = \mathcal{O}(x)$, i.e., f perfectly solves the oracle’s classification task.*

The proof of this lemma is below; at a high level, observe that given a valid distance function that satisfies Definition 5.3 we can construct a nearest neighbor classifier that perfectly matches the oracle. Thus, in general we cannot hope to have such a distance function.

Proof. We first show that if we have a distance function \mathbf{dist} that satisfies Definition 5.3, then the classification task can be perfectly solved.

Let x be an input from class y so that $\mathcal{O}(x) = y$. Let $\{x^{(i)}\}$ be any (possibly infinite) sequence of inputs so that $\mathbf{dist}(x, x^{(i)}) < \mathbf{dist}(x, x^{(i+1)})$ but so that $\mathcal{O}(x^{(i)}) = y$ for all $x^{(i)}$. Define $l_x = \lim_{i \rightarrow \infty} \mathbf{dist}(x, x^{(i)})$ as the distance to the furthest input from this class along the path $x^{(i)}$.

Assume that \mathcal{O} is not degenerate and there exists at least one input z so that $\mathcal{O}(z) \neq y$. If the problem is degenerate then it is uninteresting: *every* function \mathbf{dist} satisfies Definition 3.

Now let $\{z^{(i)}\}$ be any (possibly infinite) sequence of inputs so that $\mathbf{dist}(x, z^{(i)}) > \mathbf{dist}(x, z^{(i+1)})$ and so that $\mathcal{O}(z^{(i)}) \neq y$. Define $l_z = \lim_{i \rightarrow \infty} \mathbf{dist}(x, z^{(i)})$ as the distance to the closest input along z . But by Definition 5.3 we are guaranteed that $l_z > l_x$, otherwise there would exist an index I such that $\mathbf{dist}(x, x^{(I)}) \geq \mathbf{dist}(x, z^{(I)})$ but so that $\mathcal{O}(x) = \mathcal{O}(x^{(I)})$ and $\mathcal{O}(x) \neq \mathcal{O}(z^{(I)})$, contradicting Definition 3. Therefore for any example x , *all* examples $x^{(i)}$ that share the same class label are closer than *any* other input z that has a different class label.

From here it is easy to see that the task can be solved trivially by a 1-nearest neighbor classifier using this function \mathbf{dist} . Let $S = \{(\alpha^{(i)}, y^{(i)})\}_{i=1}^C$ contain exactly one pair (z, y) for every class. Given an arbitrary query point x , we can therefore compute the class label as $\arg \min \mathbf{dist}(x, \alpha^{(i)})$, which must be the correct label, because of the above argument: the closest example from any (incorrect) class is different than the furthest example from the correct class, and so in particular, the closest input from S *must* be the correct label.

For the reverse direction, assume we have a classifier $f(x)$ that solves the task perfectly, i.e.,

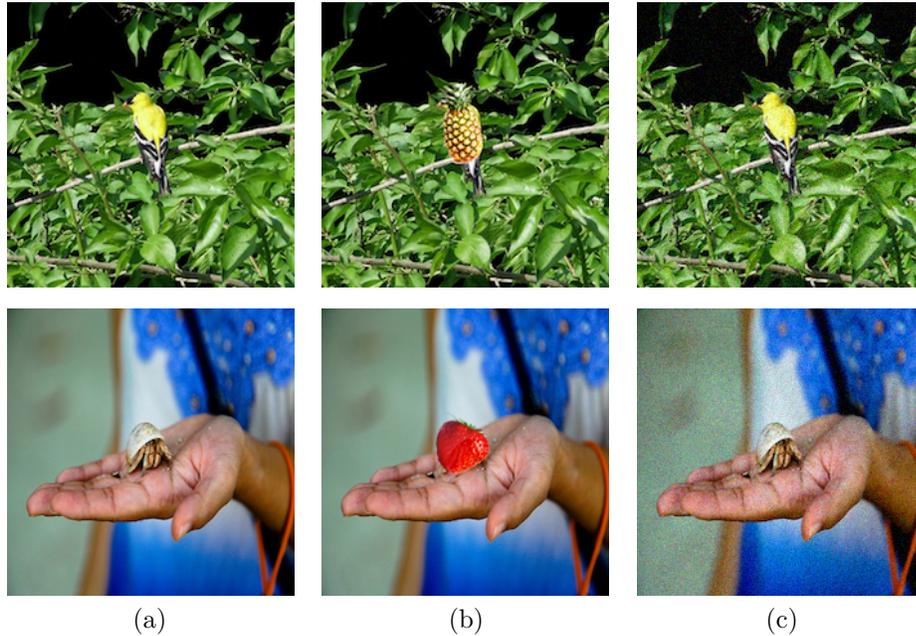


Figure 5.3: **An ℓ_p norm fails to measure semantic similarity in images.** (a) original image in the ImageNet validation set labeled as a *goldfinch* (top), *hermit crab* (bottom); (b) semantic perturbation with a ℓ_2 perturbation of 19 (respectively 22) that replaces the object of interest with a pineapple (top), strawberry (bottom). (c) random perturbation of the same ℓ_2 norm.

$f(x) = \mathcal{O}(x)$ for any $x \in \mathbb{R}^d$. Then the following distance function is aligned with the oracle.

$$\text{dist}(x, x') = \begin{cases} 0 & \text{if } f(x) = f(x') \\ 1 & \text{otherwise} \end{cases}$$

□

5.3 Generating Invariance-based Adversarial Examples on MNIST

We now empirically demonstrate and evaluate the trade-off between sensitivity-based and invariance-based adversarial examples. We propose an algorithm for generating invariance adversarial examples, and show that robustified models are disparately more vulnerable to these attacks compared to standard models. In particular, we break both *adversarially-trained* and *certifiably-robust* models on MNIST by generating invariance adversarial examples—within the models’ (possibly certified) norm bound—to which the models’ assign different labels than an ensemble of humans.

```

GenInv ( $x, y, D, \mathcal{T}$ )
   $\mathcal{S} \leftarrow \{x' : (x', y') \in D, y' \neq y\}$ 
   $D^* \leftarrow \{t(x') : t \in \mathcal{T}, x' \in \mathcal{S}\}$ 
   $\hat{x} \leftarrow \arg \min_{x' \in D^*} \|x' - x\|$ 
  return  $\hat{x}$ 

```

Algorithm 2: Meta-algorithm for finding invariance-based adversarial examples. For an input x , we find an input \hat{x} of a different class in the dataset D , that is closest to x under some set of semantics-preserving transformations. \mathcal{T} .

Why MNIST? We elect to study MNIST, the only dataset for which strong robustness to various ℓ_p bounded perturbations is attainable with current techniques [159, 222]. The dataset’s simplicity is what initially prompted the study of simple ℓ_p bounded perturbations [95]. Increasing MNIST models’ robustness to such perturbations has since become a standard benchmark [137, 159, 207, 222]. Due to the existence of models with high robustness to various ℓ_p bounded attacks, robust classification on MNIST is considered close to solved [222].

We argue that, contrary to popular belief, MNIST is far from being solved. We show why optimizing for robustness to ℓ_p bounded adversaries is not only insufficient, but actively harms the performance of the classifier against alternative invariance-based attacks.

In Section 5.3.4, we show that complex vision tasks (e.g., ImageNet) are also affected by the fundamental tradeoffs we describe. These tradeoffs are simply not apparent yet, because of our inability to train models with non-negligible robustness to any attacks on these tasks.

5.3.1 Generating Model-agnostic Invariance-based Adversarial Examples

We propose a model-agnostic algorithm for crafting invariance adversarial examples. Our attack generates minimally perturbed invariance adversarial examples that cause humans to change their classification. We then evaluate these examples against multiple models. The rationale for this approach is mainly that obtaining human labels is expensive, which encourages the use of a single attack for all models.

The high-level algorithm we use is in Algorithm 2 and described below. It is simple, albeit tailored to datasets where comparing images in pixel space is meaningful, like MNIST.³

Given an input x , the attack’s goal is to find the smallest class-changing perturbation $\hat{x} = x + \delta$ (c.f. Equation (5.2)) such that $\mathcal{O}(\hat{x}) \neq \mathcal{O}(x)$. Typically, \hat{x} is not a part of the dataset. We thus approximate \hat{x} via *semantics-preserving* transformations of other inputs. That is, for the set \mathcal{S} of inputs of a different class than x , we apply transformations \mathcal{T} (e.g., small image rotations,

³Kaushik et al. [131] consider a similar problem for NLP tasks. They ask human labelers to produce “counterfactually-augmented data” by introducing a minimal number of changes to a text document so as to change the document’s semantics.

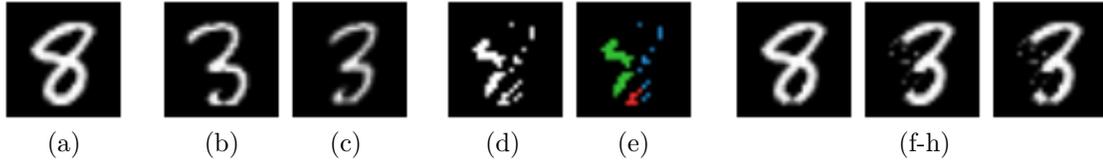


Figure 5.4: **Process for generating ℓ_0 invariant adversarial examples.** From left to right: (a) the original image of an 8; (b) the nearest training image (labeled as 3), before alignment; (c) the nearest training image (still labeled as 3), after alignment; (d) the δ perturbation between the original and aligned training example; (e) spectral clustering of the perturbation δ ; and (f-h) candidate invariance adversarial examples, selected by applying subsets of clusters of δ to the original image. (f) is a failed attempt at an invariance adversarial example. (g) is successful, but introduces a larger perturbation than necessary (adding pixels to the bottom of the 3). (h) is successful and minimally perturbed.

translations) that are known a-priori to preserve input labels. We then pick the transformed input that is closest to our target point under the considered ℓ_p metric. Below, we describe instantiations of this algorithm for the ℓ_0 and ℓ_∞ norms. Figure 5.4 visualizes the sub-steps for the ℓ_0 attack, including an extra post-processing that further reduces the perturbation size.

Measuring attack success. We refer to an invariance adversarial example as *successful* if it causes a change in the oracle’s label, i.e., $\mathcal{O}(\hat{x}) \neq \mathcal{O}(x)$. This is a model-agnostic version of Definition 5.2. In practice, we simulate the oracle by asking an ensemble of humans to label the point \hat{x} ; if more than some fraction of them agree on the label (throughout this section, 70%) and that label is different from the original, the attack is successful. Note that success or failure is independent of any machine learning model.

Generating ℓ_0 invariant adversarial examples. Assume we are given a training set D consisting of labeled example pairs (x, y) . As input our algorithm accepts an example x with oracle label $\mathcal{O}(x) = y$. Image x with label $y = 8$ is given in Figure 5.4 (a).

Define $\mathcal{S} = \{x' : (x', y') \in D, x' \neq y'\}$, the set of training examples with a different label. Now we define \mathcal{T} to be the set of transformations that we allow: rotations by up to 20 degrees, horizontal or vertical shifts by up to 6 pixels (out of 28), shears by up to 20%, and re-sizing by up to 50%.

We generate a new augmented training set $D^* = \{t(x') : t \in \mathcal{T}, x' \in \mathcal{S}\}$. By assumption, each of these examples is labeled correctly by the oracle. In our experiments, we verify the validity of this assumption through a human study and omit any candidate adversarial example that violates this assumption. Finally, we search for

$$\hat{x} = \arg \min_{\hat{x} \in D^*} \|\hat{x} - x\|_0.$$

By construction, we know that x and \hat{x} are similar in pixel space but have a different label. Figure 5.4 (b-c) show this step of the process. Next, we introduce a number of refinements to make \hat{x} be “more

similar” to x . This reduces the ℓ_0 distortion introduced to create an invariance-based adversarial example—compared to directly returning \hat{x} as the adversarial example.

First, we define $\delta = |x - \hat{x}| > 1/2$ where the absolute value and comparison operator are taken element-wise. Intuitively, δ represents the pixels that substantially change between \hat{x} and x . We choose $1/2$ as an arbitrary threshold representing how much a pixel changes before we consider the change “important”. This step is shown in Figure 5.4 (d). Along with δ containing the *useful* changes that are responsible for changing the oracle class label of x , it also contains irrelevant changes that are superficial and do not contribute to changing the oracle class label. For example, in Figure 5.4 (d) notice that the green cluster is the only semantically important change; both the red and blue changes are not necessary.

To identify and remove the superficial changes, we perform spectral clustering on δ . We compute $\delta^{(i)}$ by enumerating all possible subsets of clusters of pixel regions. This gives us many possible *potential* adversarial examples $\hat{x}^{(i)} = x + \delta^{(i)}$. Notice these are only potential because we may not actually have applied the necessary change that actually modifies the class label.

We show three of the eight possible candidates in Figure 5.4. In order to alleviate the need for human inspection of each candidate $\hat{x}^{(i)}$ to determine which of these potential adversarial examples is actually misclassified, we follow an approach from Defense-GAN [219] and the Robust Manifold Defense [113]: we take the generator from a GAN and use it to assign a likelihood score to the image. We make one small refinement, and use an AC-GAN [172] and compute the class-conditional likelihood of this image occurring. This process reduces ℓ_0 distortion by 50% on average.

As a small refinement, we find that initially filtering D by removing the 20% least-canonical examples makes the attack succeed more often.

Generating ℓ_∞ invariant adversarial examples. Our approach for generating ℓ_∞ invariant examples follows similar ideas as for the ℓ_0 case, but is conceptually simpler as the perturbation budget can be applied independently for each pixel (our ℓ_∞ attack is however less effective than the ℓ_0 one, so further optimizations may prove useful).

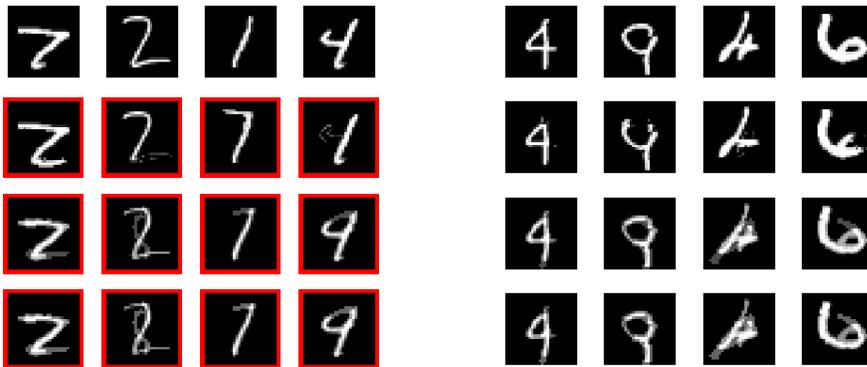
We build an augmented training set D^* as in the ℓ_0 case. Instead of looking for the closest nearest neighbor for some example x with label $\mathcal{O}(x) = y$, we restrict our search to examples $\hat{x} \in D^*$ with specific target labels y^* , which we’ve empirically found to produce more convincing examples (e.g., we always match digits representing a 1, with a target digit representing either a 7 or a 4). We then simply apply an ℓ_∞ bounded perturbation to x by interpolating with \hat{x} , so as to minimize the distance between x and the chosen target example \hat{x} .

5.3.2 Evaluation

Attack analysis. We generate 100 invariance adversarial examples on inputs randomly drawn from the MNIST test set, for both the ℓ_0 and ℓ_∞ norms. Our attack is slow, with the alignment

Table 5.1: Success rate of invariance adversarial examples in causing humans to switch their classification.

Attack Type	Success Rate
Clean Images	0%
ℓ_0 Attack	55%
$\ell_\infty, \varepsilon = 0.3$ Attack	21%
$\ell_\infty, \varepsilon = 0.3$ Attack (manual)	26%
$\ell_\infty, \varepsilon = 0.4$ Attack	37%
$\ell_\infty, \varepsilon = 0.4$ Attack (manual)	88%

Figure 5.5: **Invariance-based adversarial examples on MNIST.** Top to bottom: original images and our ℓ_0 , ℓ_∞ at $\varepsilon = 0.3$ and ℓ_∞ at $\varepsilon = 0.4$ invariance adversarial examples. (left) successful attacks; (right) failed attack attempts.

process taking (amortized) minutes per example. We performed no optimizations of this process and expect it could be improved. The mean ℓ_0 distortion of successful examples is 25.9 (with a median of 25). The ℓ_∞ attack always uses the full budget of either $\varepsilon = 0.3$ or $\varepsilon = 0.4$ and runs in a similar amount of time.

Human study. We conducted a human study to evaluate whether our invariance adversarial examples are indeed successful, i.e., whether humans agree that the label has been changed. We also hand-crafted 50 invariance adversarial examples for the ℓ_0 and ℓ_∞ norm. The process was quite simple: we built an image editor that lets us change images at a pixel level under an ℓ_p constraint. One author then modified 50 random test examples in the way that they perceived as changing the underlying class. We presented all these invariance examples to 40 human evaluators. Each evaluator classified 100 digits, half of which were unmodified MNIST digits, and the other half were sampled randomly from our ℓ_0 and ℓ_∞ invariance adversarial examples.

Table 5.2: **Agreement between models and humans on invariance adversarial examples.** Shows model accuracy with respect to the oracle human labelers on the subset of examples where the human-obtained oracle label is different from the test label. Models which are *more* robust to *perturbation* adversarial examples (such as those trained with adversarial training) tend to agree with humans **less often** on *invariance-based* adversarial examples. Values denoted with an asterisks * violate the perturbation threat model of the defense and should not be taken to be attacks. When the model is *wrong*, it failed to classify the input as the new oracle label.

Model: ¹	Undefended	ℓ_0 Sparse	Binary-ABS	ABS	ℓ_∞ PGD ($\varepsilon = 0.3$)	ℓ_2 PGD ($\varepsilon = 2$)
Clean	99%	99%	99%	99%	99%	99%
ℓ_0	80%	38%	47%	58%	56%*	27%*
$\ell_\infty, \varepsilon = 0.3$	33%	19%*	0%	14%	0%	5%*
$\ell_\infty, \varepsilon = 0.4$	51%	27%*	8%	18%	16%*	19%*

¹ ℓ_0 Sparse: [9]; ABS and Binary-ABS: [222]; ℓ_∞ PGD and ℓ_2 PGD: [159]

Results. Of 100 clean (unmodified) test images, 98 are labeled identically by *all* human evaluators. The other 2 images were labeled identically by over 90% of evaluators.

Our ℓ_0 attack is highly effective: For 55 of the 100 examples at least 70% of human evaluators labeled it the same way, with a different label than the original test label. Humans only agreed with the original test label (with the same 70% threshold) on 34 of the images, while they did not form a consensus on 18 examples. The simpler ℓ_∞ attack is less effective: with a distortion of 0.3 the oracle label changed 21% of the time and with 0.4 the oracle label changed 37% of the time. The manually created ℓ_∞ examples with distortion of 0.4 were highly effective however: for 88% of the examples, at least 70% assigned the same label (different than the test set label). We summarize results in Table 5.1. In Figure 5.5 we show sample invariance adversarial examples.

To simplify the analysis below, we split our generated invariance adversarial examples into two sets: the successes and the failures, as determined by whether the plurality decision by humans was different than or equal to the original label. We only evaluate models on those invariance adversarial examples that caused the humans to switch their classification.

Model evaluation. Given oracle ground-truth labels for each of the images (as decided by humans), we report how often models agree with the human-assigned label. Table 5.2 summarizes this analysis. For the invariance adversarial examples, we report model accuracy only on *successful* attacks (i.e., those where the human oracle label changed between the original image and the modified image).⁴ For these same models, Table 5.3 reports the “standard” robust accuracy for sensitivity-based adversarial examples, i.e., in the sense of Equation (5.1).

The models which empirically achieve the highest robustness against ℓ_0 perturbations (in the

⁴It may seem counter-intuitive that our ℓ_∞ attack with $\varepsilon = 0.3$ appears stronger than the one with $\varepsilon = 0.4$. Yet, given two successful invariance examples (i.e., that both change the human-assigned label), the one with lower distortion is expected to change a model’s output less often, and is thus a stronger invariance attack.

Table 5.3: **Model accuracy on sensitivity-based adversarial examples.** Shows robust model accuracy with respect to the original MNIST labels under different threat models. To measure ℓ_0 robustness, we use the PointwiseAttack of [222] repeated 10 times, with $\varepsilon = 25$. For ℓ_∞ robustness, we use PGD with 100 iterations for $\varepsilon = 0.3$ and $\varepsilon = 0.4$. For the ABS and Binary-ABS models, we report the number from [222], for PGD combined with stochastic gradient estimation.

Model:	Undefended	ℓ_0 Sparse	Binary-ABS	ABS	ℓ_∞ PGD ($\varepsilon = 0.3$)	ℓ_2 PGD ($\varepsilon = 2$)
ℓ_0 Attack ($\varepsilon = 25$)	0%	45%	63%	43%	0%	40%
ℓ_∞ Attack ($\varepsilon = 0.3$)	0%	8%	77%	8%	92%	1%
ℓ_∞ Attack ($\varepsilon = 0.4$)	0%	0%	60%	0%	7%	0%

sense of Equation (5.1)) are the ℓ_0 Sparse classifier of Bafna et al. [9], the Binary-ABS model of Schott et al. [222], and the ℓ_2 PGD adversarially trained model (see Table 5.3 for a comparison of the robustness of these models). Thus, these are the models that are most *invariant* to perturbations of large ℓ_0 norm. We find that these are the models that achieve the lowest accuracy—as measured by the human labelers—on our invariance examples. Moreover, all robust models perform much worse than an undefended ResNet-18 model on our invariance attacks. This includes models such as the ℓ_∞ PGD adversarially trained model, which do not explicitly aim at worst-case robustness against ℓ_0 noise. Thus, we find that models that were designed to reduce excessive sensitivity to certain non-semantic features, become excessively invariant to other features that are semantically meaningful.

Similarly, we find that models designed for ℓ_∞ robustness (Binary-ABS and ℓ_∞ PGD) also fare the worst on our ℓ_∞ invariance adversarial examples. Overall, all robust models do worse than the undefended baseline. The results are consistent for attacks with $\varepsilon = 0.3$ and with $\varepsilon = 0.4$, the latter being more successful in changing human labels.

Note that the Binary-ABS defense of [222] boasts 60% (empirical) robust accuracy on ℓ_∞ attacks with $\varepsilon = 0.4$ (see [222]). Yet, on our our invariance examples that satisfy this perturbation bound, the model actually disagrees with the human labelers 92% of the time, and thus achieves only 8% true accuracy on these examples. Below, we make a similar observation for a *certified* defense.

5.3.3 Trading Perturbation-robustness for Invariance-robustness

To better understand how robustness to sensitivity-based adversarial examples influences robustness to invariance attacks, we evaluate a range of adversarially-trained models on our invariance examples.

Setup. We trained ℓ_∞ PGD models with $\varepsilon \in [0, 0.4]$ and ℓ_1 PGD models (as a proxy for ℓ_0 robustness) with $\varepsilon \in [0, 15]$.

We use the same architecture as [159]. We train each model for 10 epochs with Adam and a learning rate of 10^{-3} reduced to 10^{-4} after 5 epochs (with a batch size of 100). To accelerate

Table 5.4: **Robust accuracy as a function of perturbation size during training.** Models are trained against attacks of increasing magnitude. The models trained on ℓ_∞ attacks (left) are evaluated against ℓ_∞ PGD with $\varepsilon \in \{0.3, 0.4\}$. The models trained on ℓ_1 attacks (right) are evaluated against the ℓ_0 Pointwise attack [222]. Accuracy is measured with respect to the original MNIST label.

Attack	ε for ℓ_∞ PGD training				Attack	ε for ℓ_1 PGD training		
	0.1	0.2	0.3	0.4		5	10	15
PGD $\varepsilon = 0.3$	0%	6%	92%	93%	ℓ_0 PointwiseAttack ($\varepsilon = 25$)	41%	59%	65%
PGD $\varepsilon = 0.4$	0%	0%	7%	90%				

convergence, we train against a weaker adversary in the first epoch (with 1/3 of the perturbation budget). For training, we use PGD with 40 iterations for ℓ_∞ and 100 iterations for ℓ_1 . For ℓ_∞ PGD, we choose a step-size of $2.5 \cdot \varepsilon/k$, where k is the number of attack iterations. For the models trained with ℓ_1 PGD, we use the Sparse ℓ_1 Descent Attack of Tramèr and Boneh [250], with a sparsity fraction of 99%.

Results. We first verify, in Table 5.4 that training against larger perturbations results in a monotonic increase in adversarial robustness, in the sense of Equation (5.1).

We then evaluate these models against respectively the ℓ_∞ and ℓ_0 invariance examples. Figure 5.6 shows that robustness to larger perturbations leads to higher vulnerability to invariance-based examples.

Interestingly, while sensitivity-based robustness does not generalize beyond the norm-bound on which a model is trained (e.g., a model trained on PGD with $\varepsilon = 0.3$ achieves very little robustness to PGD with $\varepsilon = 0.4$ [159]), excessive invariance does generalize (e.g., a model trained on PGD with $\varepsilon = 0.2$ is more vulnerable to our invariance attacks with $\varepsilon \geq 0.3$ compared to an undefended model).

Breaking certified defenses. Our invariance attacks even constitute a *break* of some certified defenses. For example, Zhang et al. [286] develop a defense which *proves* that the accuracy on the test set is at least 87% under ℓ_∞ perturbations of size $\varepsilon = 0.4$. When we run their pre-trained model on all 100 of our $\varepsilon = 0.4$ invariance adversarial examples (i.e., not just the successful ones) we find it has a 96% “accuracy” (i.e., it matches the original test label 96% of the time). However, when we look at the agreement between this model’s predictions with the new labels assigned by the human evaluators, the model’s accuracy is just 63%.

Thus, while the proof in the paper is *mathematically correct* it does not actually deliver 87% robust accuracy under ℓ_∞ attacks with $\varepsilon = 0.4$: humans change their classification for many of these perturbations. Worse, for the 50 adversarial examples we crafted by hand, the model *disagrees* with the human ensemble 88% of the time: it has just 12% accuracy.

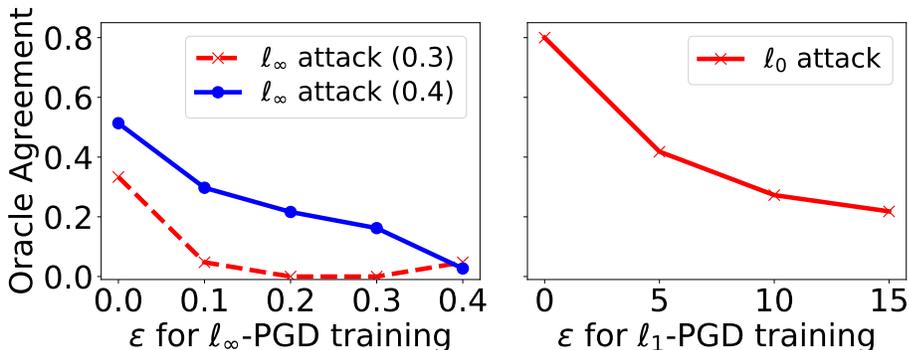


Figure 5.6: **Higher noise-robustness leads to higher vulnerability to invariance attacks.** (left) For models trained with ℓ_∞ PGD, a higher bound $\epsilon \in [0, 0.4]$ implies lower accuracy on ℓ_∞ bounded invariance examples. (right) Models trained with ℓ_1 PGD evaluated on the ℓ_0 invariance attack.

5.3.4 Natural Images

While our experiments are on MNIST, similar phenomena may arise in other vision tasks. Figure 5.3 shows two perturbations of ImageNet images: the rightmost perturbation is imperceptible and thus classifiers should be robust to it. Conversely, the middle image was semantically changed, and classifiers should be sensitive to such changes. Yet, the ℓ_2 norm of both perturbations is the same. Hence, enforcing robustness to ℓ_2 noise of some fixed size ϵ will necessarily result in a classifier that is either sensitive to the changes on the right, or invariant to the changes in the middle image. Such a phenomenon will necessarily arise for any image dataset that contains small objects, as perturbations of small ℓ_2 magnitude will be sufficient to occlude the object, thereby changing the image semantics.

This distance-oracle misalignment extends beyond the ℓ_2 norm. For instance, Co et al. [51] show that a perturbation of size $16/255$ in ℓ_∞ can suffice to give an image of a cat the appearance of a shower curtain print, which are both valid ImageNet classes. Yet, a *random* perturbation of the same magnitude is semantically meaningless.

On CIFAR-10, some recent defenses are possibly already overly invariant. For example, Shaeiri et al. [224] and Panda et al. [190] aim to train models that are robust to ℓ_∞ perturbations of size $\epsilon = 32/255$. Yet, Tsipras et al. [259] show that perturbations of that magnitude can be semantically meaningful and can be used to effectively *interpolate* between CIFAR-10 classes. The approach taken by Tsipras et al. [259] to create these perturbations, which is based on a model with robustness to very small ℓ_∞ noise, may point towards an efficient way of automating the generation of invariance attacks for tasks beyond MNIST. The work of Sharif et al. [229] also shows that “small” ℓ_∞ noise (of magnitude $25/255$) can reliably fool human labelers on CIFAR-10.

5.4 The Overly-robust Features Model

The experiments in Section 5.3 show that models can be robust to perturbations large enough to change an input’s semantics. Taking a step back, it is not obvious why training such classifiers is possible, i.e., why does excessive invariance not harm regular accuracy. To understand the learning dynamics of these overly-robust models, we ask two questions:

1. Can an overly-robust model fit the training data?
2. Can such a model generalize (robustly) to test data?

5.4.1 Formal Model and Analysis

For simplicity, we assume that for every point $(x, y) \sim \mathcal{D}$, the closest point \hat{x} (under the chosen norm) for which $\mathcal{O}x\hat{x} \neq y$ is at a constant distance ε^* . We train a model f to have low robust error (as in Equation (5.1)) for perturbations of size $\varepsilon > \varepsilon^*$. This model is thus overly-robust.

We first ask under what conditions f may have low robust training error. A necessary condition is that there do not exist training points $(x^{(i)}, y^{(i)}), (x^{(j)}, y^{(j)})$ such that $y^{(i)} \neq y^{(j)}$ and $\|x^{(i)} - x^{(j)}\| \leq \varepsilon$. As ε is larger than the inter-class distance, the ability to fit an overly robust model thus relies on the training data not being fully representative of the space to which the oracle assigns labels. This seems to be the case in MNIST: as the dataset consists of centered, straightened and binarized digits, even an imaginary infinite-sized dataset might not contain our invariance adversarial examples.

The fact that excessive robustness *generalizes* (as provably evidenced by the model of Zhang et al. [286]) points to a deeper issue: there must exist *overly-robust* and *predictive* features in the data—that are not aligned with human perception. This mirrors the observations of [115], who show that excessive sensitivity is caused by non-robust yet predictive features. On MNIST, our experiments confirm the existence of overly-robust generalizable features.

We formalize these observations using a simple classification task inspired by [259]. We consider a binary task where inputs $x \in \mathbb{R}^{d+2}$ are sampled from a distribution \mathcal{D}_k^* with parameter k :

$$z \stackrel{\text{u.a.r.}}{\sim} \{-1, 1\}, \quad x_1 = z/2$$

$$x_2 = \begin{cases} +z & \text{w.p. } \frac{1+1/k}{2} \\ -z & \text{w.p. } \frac{1-1/k}{2} \end{cases}, \quad x_3, \dots, x_{d+2} \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}\left(\frac{z}{\sqrt{d}}, k\right).$$

Here $\mathcal{N}(\mu, \sigma^2)$ is a normal distribution and $k > 1$ is a constant chosen so that only feature x_1 is strongly predictive of the latent variable z (e.g., $k = 100$ so that x_2, \dots, x_{d+2} are almost uncorrelated with z). The oracle is defined as $\mathcal{O}(x) = \text{sign}(x_1)$, i.e., feature x_1 fully defines the oracle’s class label, and other features are nearly uncorrelated with it. Note that the oracle’s labels are robust under any ℓ_∞ noise with norm strictly below $\varepsilon = 1/2$.

We model the collection of “sanitized” and labeled datasets from a data distribution as follows: the semantic features (i.e., x_1) are preserved, while “noise” features have their variance reduced (e.g., because non-standard inputs are removed). Sanitization thus enhances “spurious correlations” [115, 124] between non-predictive features and class labels.⁵ We further assume that the data labeling process introduces some small label noise.⁶ Specifically, the labeled data distribution \mathcal{D} on which we train and evaluate classifiers is obtained by sampling x from a sanitized distribution $\mathcal{D}_{1+\alpha}^*$ (for a small constant $\alpha > 0$) where features x_2, \dots, x_{d+2} are strongly correlated with the oracle label. The label y is set to the correct oracle label with high probability $1 - \beta$:

$$x \sim \mathcal{D}_{1+\alpha}^*, \quad y = \begin{cases} +\mathcal{O}(x) & \text{w.p. } 1 - \beta \\ -\mathcal{O}(x) & \text{w.p. } \beta \end{cases}.$$

The consequences of this data sanitization are two-fold:

1. A standard classifier (that maximizes accuracy on \mathcal{D}) agrees with the oracle with probability at least $1 - \beta$, but is vulnerable to ℓ_∞ perturbations of size $\varepsilon = O(d^{-1/2})$.
2. There is an overly-robust model that only uses feature x_2 and has robust accuracy $1 - \alpha/2$ on \mathcal{D} for ℓ_∞ noise of size $\varepsilon = 0.99$. This classifier is vulnerable to invariance attacks as the oracle is not robust to such perturbations.

A standard classifier is vulnerable to adversarial examples. We first show that this sanitization introduces spurious weakly robust features. Standard models trained on \mathcal{D} are thus vulnerable to sensitivity-based adversarial examples.

Lemma 5.5. *Let $f(x)$ be the Bayes optimal classifier on \mathcal{D} . Then f agrees with the oracle \mathcal{O} with probability at least $1 - \beta$ over \mathcal{D} but with 0% probability against an ℓ_∞ adversary bounded by some $\varepsilon = O(d^{-1/2})$.*

Proof. The first part of the lemma, namely that f agrees with the oracle \mathcal{O} with probability at least $1 - \beta$ follows from the fact that for $(x, y) \sim \mathcal{D}$, $\mathbf{sign}(x_1) = y$ with probability $1 - \beta$, and $\mathcal{O}(x) = \mathbf{sign}(x_1)$. So a classifier that only relies on feature x_1 achieves $1 - \beta$ accuracy. To show that the Bayes optimal classifier for \mathcal{D} has adversarial examples, note that this classifier is of the form

$$\begin{aligned} f(x) &= \mathbf{sign}(w^\top x + C) \\ &= \mathbf{sign}(w_1 \cdot x_1 + w_2 \cdot x_2 + \sum_{i=3}^{d+2} w_i \cdot x_i + C), \end{aligned}$$

⁵In digit classification for example, the number of pixels above 1/2 is a feature that is presumably very weakly correlated with the class 8. In the MNIST dataset however, this feature is fairly predictive of the class 8 and robust to ℓ_∞ noise of size $\varepsilon = 0.4$.

⁶This technicality avoids that classifiers on \mathcal{D} can trivially learn the oracle labeling function. Alternatively, we could define feature x_1 so that is is hard to learn for certain classes of classifiers.

where w_1, w_2, C are constants, and $w_i = O(1/\sqrt{d})$ for $i \geq 3$. Thus, a perturbation of size $O(1/\sqrt{d})$ applied to features x_3, \dots, x_{d+2} results in a change of size $O(1)$ in $w^T x + C$, which can be made large enough to change the output of f with arbitrarily large probability. As perturbations of size $O(1/\sqrt{d})$ cannot change the oracle’s label, they can reduce the agreement between the classifier and oracle to 0%. \square

An overly-robust model is vulnerable to invariance attacks. We further show that there exists an overly-robust classifier on \mathcal{D} that is vulnerable to invariance adversarial examples:

Lemma 5.6. *Let $f(x) = \text{sign}(x_2)$. This classifier has accuracy above $1 - \alpha/2$ on \mathcal{D} , even against an ℓ_∞ adversary bounded by $\varepsilon = 0.99$. Under such large perturbations, f agrees with the oracle with probability 0%.*

Proof. The robust accuracy of f follows from the fact that $f(x)$ cannot be changed by any perturbation of ℓ_∞ norm strictly below 1, and that for $(x, y) \sim \mathcal{D}$, we have $x_2 = y$ with probability $\frac{1+1/(1+\alpha)}{2} \geq 1 - \alpha/2$. For any $(x, y) \sim \mathcal{D}$, note that a perturbation of ℓ_∞ norm above 1/2 can always flip the oracle’s label. So we can always find a perturbation δ such that $\|\delta\|_\infty \leq 0.99$ and $f(x + \delta) \neq \mathcal{O}(x + \delta)$. \square

The role of data augmentation. This simple task suggests a natural way to prevent the training of overly robust models. If *prior knowledge* about the task suggests that classification should be invariant to features $x_2 \dots, x_{d+2}$, then enforcing these invariances would prevent a model from being robust to excessively large perturbations.

A standard way to enforce invariances is via data augmentation. In the above binary task, augmenting the training data by randomizing over features x_2, \dots, x_{d+2} would force the model to rely on the only truly predictive feature, x_1 .

5.4.2 Experiments

We experimented with aggressive data-augmentation on MNIST. For values of $\varepsilon \in [0, 0.4]$, we train models with an adversary that rotates and translates inputs by a small amount and then adds ε -bounded ℓ_∞ noise. This attacker mirrors the process we use to generate invariance adversarial examples in Section 5.3. Thus, we expect it to be hard to achieve robustness to attacks with large ε on this dataset, as this requires the model to correctly classify inputs that humans consider mislabeled.

Setup. For $\varepsilon \in \{0, 0.1, 0.2, 0.3, 0.4\}$, we train a model against an attack that first rotates and translates an input (using the default parameters from [71]) and then adds noise of ℓ_∞ norm bounded by ε to the transformed input. For training, we sample 10 spatial transformations at random for each input, apply 40 steps of ℓ_∞ PGD to each transformed input, and retain the strongest adversarial

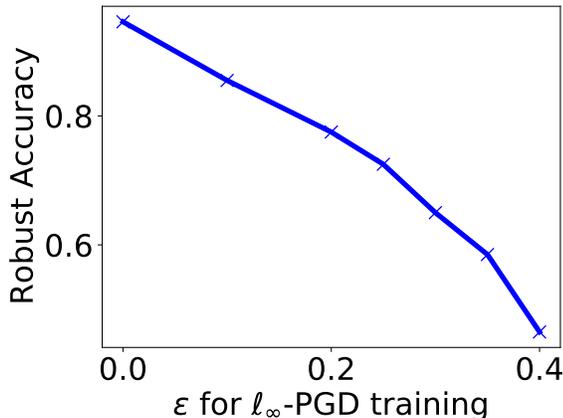


Figure 5.7: **Robust accuracy against an affine adversary.** Shows the accuracy of models trained and evaluated on an adversary combining a small spatial data augmentation (rotation + translation) with an ℓ_∞ perturbation bounded by ϵ .

example. At test time, we enumerate all possible spatial transformations for each input, and apply 100 steps of PGD to each.

When training against an adversary with $\epsilon \geq 0.25$, a warm-start phase is required to ensure training converges. That is, we first trained a model against an $\epsilon = 0.2$ adversary, and then successively increases ϵ by 0.05 every 5 epochs.

Results. Figure 5.7 confirms our intuition: as ϵ grows, it becomes harder to learn a model that is invariant to both spatial data augmentation and ℓ_∞ noise. We further find that the models trained with data augmentation agree more often with human labelers on our invariance attacks (see Figure 5.8). Yet, even with data augmentation, models trained against large ℓ_∞ perturbations still perform worse than an undefended model. This simple experiment thus demonstrates that while data-augmentation (over truly invariant features) can help in detecting or preventing excessive invariance to semantic features, even though it is not currently sufficient for training models that resist both sensitivity-based and invariance-based attacks.

5.5 Discussion

Our results show that solely focusing on robustness to sensitivity-based attacks is insufficient, as mis-specified bounds can cause vulnerability to invariance-based attacks.

On ℓ_p norm evaluations. Our invariance attacks are able to find points within the ℓ_p ball in which state-of-the-art classifiers are (provably) robust. This highlights the need for a more careful selection of perturbation bounds when measuring robustness to adversarial examples. At the same

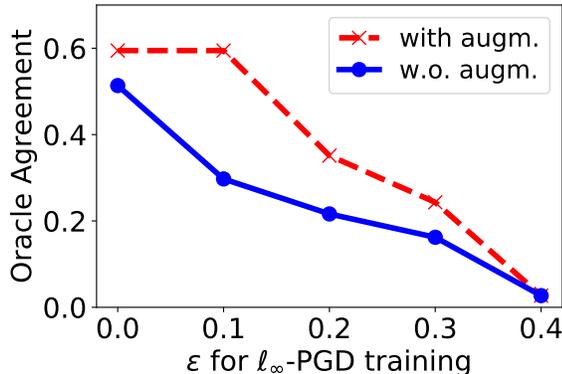


Figure 5.8: **Model-human agreement on successful invariance adversarial examples.** The invariance examples are of ℓ_∞ norm bounded by 0.4. Models trained with data augmentation agree more often with humans, and are thus more sensitive to semantically-meaningful changes.

time, Figure 5.6 shows that even promoting robustness within conservative bounds causes excessive invariance. The tradeoff explored in Section 5.2 suggests that aiming for robustness against ℓ_p bounded attacks may be inherently futile for making models robust to arbitrary adversarial examples.

Trading sensitivity and invariance. We show that models that are robust to small perturbations make excessively invariant decisions and are thus vulnerable to other attacks.

Interestingly, Engstrom et al. [70] show an opposite effect for models’ internal representations. Denoting the logit layer of a model as $z(x)$, they show that for robust models it is hard to find inputs x, \hat{x} such that $\mathcal{O}(x) \neq \mathcal{O}(\hat{x})$ and $z(x) \approx z(\hat{x})$. Conversely, Sabour et al. [217] and Jacobsen et al. [120] show that excessive invariance of feature layers is common in non-robust models. These observations are orthogonal to ours as we study invariances in a model’s classification layer, and for bounded perturbations. As we show in Section 5.2, robustness to large perturbations under a norm that is misaligned with human perception necessarily causes excessive invariance of the model’s classifications (but implies nothing about the model’s feature layers).

Increasing model robustness to ℓ_p noise also leads to other tradeoffs, such as reduced accuracy [259] or reduced robustness to other small perturbations [130, 250, 281].

5.6 Conclusion

We have introduced and studied a fundamental tradeoff between two types of adversarial examples, that stem either from excessive sensitivity or invariance of a classifier. This tradeoff is due to an inherent misalignment between simple robustness notions and a task’s true perceptual metric. We have demonstrated that defenses against ℓ_p bounded perturbations on MNIST promote invariance to semantic changes. Our attack exploits this excessive invariance by changing image semantics while

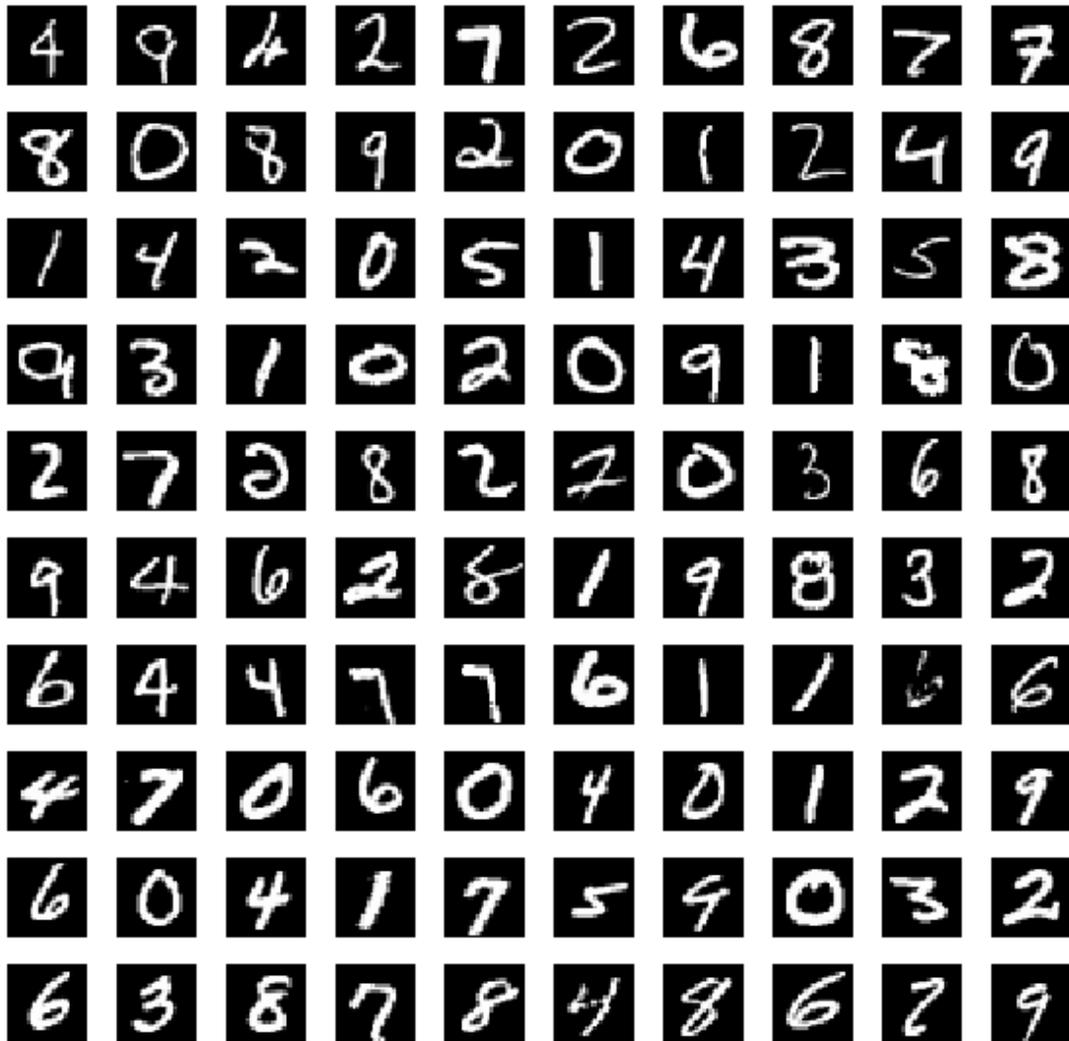
preserving model decisions. For adversarially-trained and certified defenses, our attack can reduce a model's true accuracy to random guessing. Finally, we have studied the tradeoff between sensitivity and invariance in a theoretical setting where excessive invariance can be explained by the existence of overly-robust features.

Our results highlight the need for a more principled approach in selecting meaningful robustness bounds and in measuring progress towards more robust models.

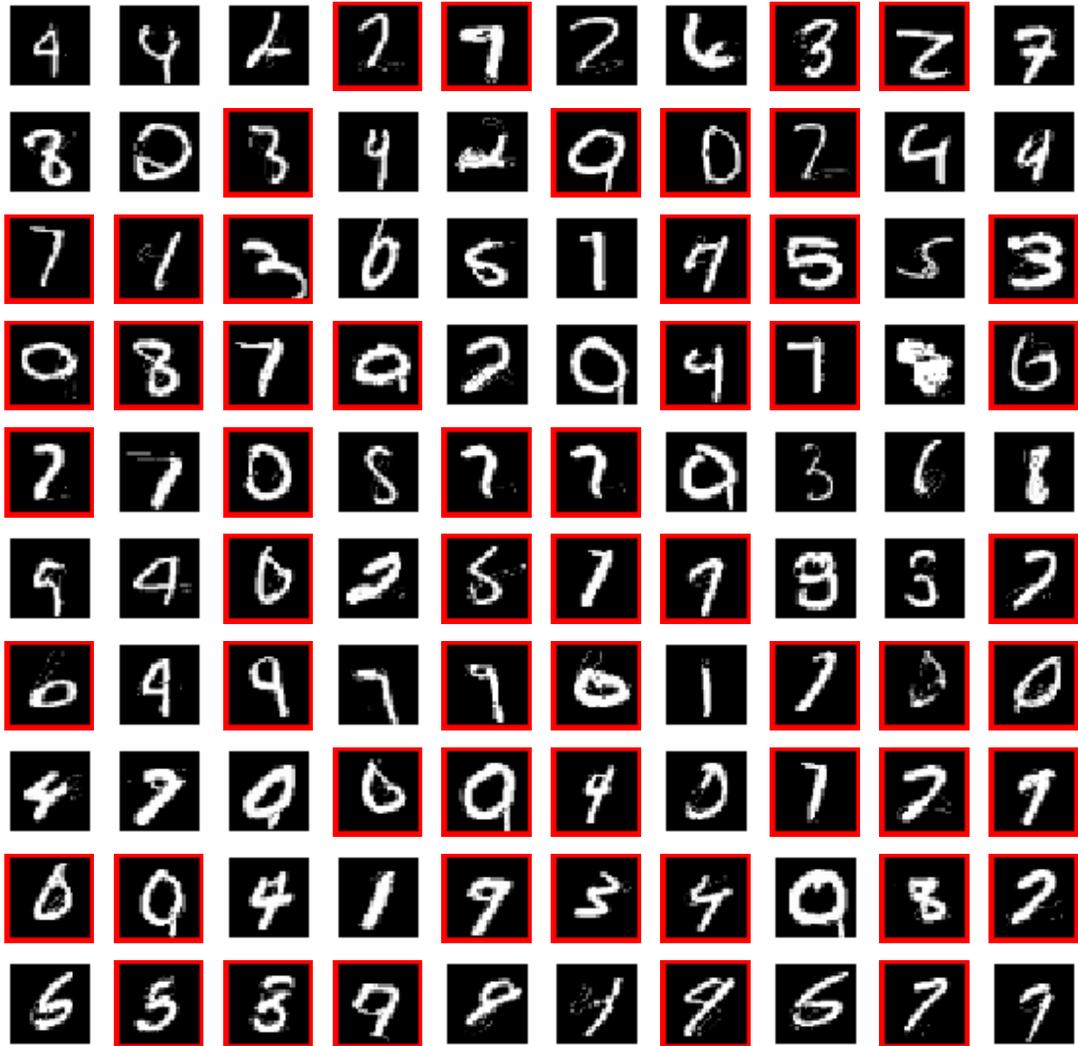
5.7 Complete Set of Invariance Adversarial Examples

Below we give the 100 randomly-selected test images along with the invariance adversarial examples that were shown during the human study.

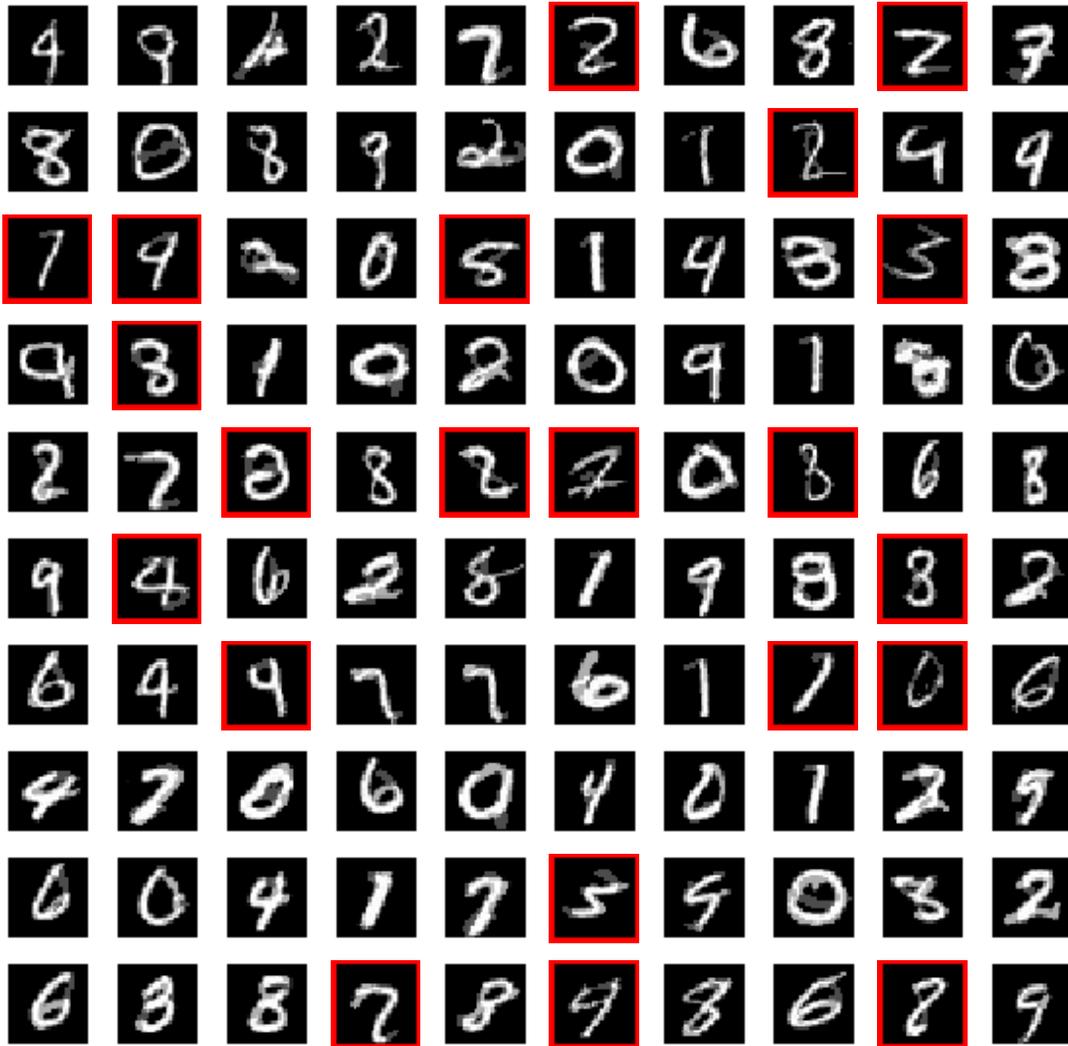
Original images.



All ℓ_0 invariance adversarial examples.



All ℓ_∞ invariance adversarial examples ($\epsilon = 0.3$).



All ℓ_∞ invariance adversarial examples ($\epsilon = 0.4$).



Part II

Privacy-Preserving Machine Learning

In the second part of this dissertation, we focus on the problem of preserving *user privacy* in machine learning systems. Compared to the notion of robust learning which we covered in the first part of this dissertation, the notion of private learning is seemingly easier to define. Indeed, robustness to adversarial examples is inherently tied to human perception, and thus hard to formally characterize. In contrast, the notions of privacy that we introduce hereafter are agnostic to the particular learning task at hand, and instead consider generic mathematical bounds on the amount of information that a machine learning system leaks about users’ data.

From a privacy perspective, a machine learning system can generically be viewed as a protocol that evaluates some *function* `func` over inputs `inpi` belonging to m different parties, to produce outputs `outi` for each party:

$$\text{out}_1, \dots, \text{out}_m \leftarrow \text{func}(\text{inp}_1, \dots, \text{inp}_m).$$

For example, when m users jointly train a model on their own data, each user’s input `inpi` is a training set $D_{\text{train}}^{(i)}$ (and possibly some randomness $r^{(i)}$), and the function `func` outputs a trained model $\text{out}_1 = \dots = \text{out}_m := f$. As a second example, when a client outsources predictions to a remote service (i.e., $m = 2$), the client’s input is its evaluation data, $\text{inp}_1 := \{x\}$, the remote service’s input is the trained model, $\text{inp}_2 := \{f\}$, and the function `func` computes the model prediction for the client, i.e., $\text{out}_1 := f(x)$ (the service provider receives no output, $\text{out}_2 := \perp$).

What does it mean for such a protocol to be *private*? Intuitively, each protocol participant would like to enjoy the benefits from the function’s output, whilst preventing other parties from learning something about their input `inpi`. This intuition can be mathematically formalized in different ways, which leads to orthogonal and complementary types of privacy protection.

Secure computation. One way of formalizing privacy originated in the literature on *secure computation* in cryptography [12, 91, 279]. Secure computation relates to the privacy of the *computation* of the function `func`. Intuitively, a protocol for computing the function `func` is secure if any (computationally bounded) adversary learns nothing more than it could have also learned if the entire computation had been performed by an “ideal trusted party”. This ideal party collects all parties’ inputs `inpi`, computes `func`, and distributes the outputs `outi`. Thus, secure computation asks that the adversary learns nothing more about a user’s input `inpi`, than what can be inferred from the adversary’s own inputs and outputs, $(\text{inp}_j, \text{out}_j)$.

A remarkable result of Goldreich et al. [91] states that *any* function that can be efficiently computed can also be efficiently computed securely (under standard cryptographic assumptions). Thus, the privacy of the *computation* of a function `func` can always be guaranteed.

Instead of relying on purely cryptographic techniques, which leads to large performance overheads, Trusted Execution Environments (TEEs) can present a more pragmatic approach to secure

computation. A TEE is a processor that relies on hardware and software protections to isolate its execution from other applications, and that has the ability to attest to remote parties that it is executing a given piece of software. TEEs enable a simple and efficient solution to many secure computation problems: one party hosts a TEE that attests that it is running the function `func`, and each party sends their input encrypted to this TEE. The TEE then decrypts all inputs, runs the function, and sends back the encrypted outputs.

The privacy guarantee offered by secure computation is sufficient for outsourcing the *evaluation* of machine learning models: if a client’s predictions were outsourced to an ideal party, the adversary would learn *nothing* about the user’s input. Prior work has shown how to construct secure computation protocols tailored to the task of machine learning predictions, in particular for neural networks, using either cryptographic techniques [126, 174] or TEEs [45, 103, 111, 183].

The situation for *training* machine learning models is more delicate, as secure computation guarantees nothing about the privacy *of the function `func` itself*. To illustrate, consider a trivial example where the function `func` is the identity function, i.e.,

$$\text{out}_1 = \dots \text{out}_m := \{\text{inp}_1, \dots, \text{inp}_m\}.$$

This is certainly an efficiently computable function, and thus we can build a protocol that computes it “securely”. But this protocol would be trivial: simply collect and output every party’s data. This protocol is private in the sense of secure computation, as the computation of the function reveals nothing more than the function’s output. Yet, such a protocol clearly does not preserve the privacy of users’ data.

While this example may seem extreme, machine learning algorithms are closer to this trivial function than we might think. Consider for instance the function that learns a nearest-neighbor classifier: the output function f is simply a database that contains all the training data. Similarly, in a Support Vector Machine [54], the learned model’s support vectors (which are used to compute predictions on new data) are also training data points. Finally, while deep neural networks do not seem to encode their training data in such an explicit fashion, they certainly do encode partial information about the training data points in the learned parameters. Indeed, prior work has shown how to *extract* training data by interacting with a machine learning model [31, 32, 81]. Learning algorithms thus call for a stronger notion of privacy, than privacy of the computation alone.

Differential privacy. A private learning algorithm should ideally prevent that the learned model itself leaks information about each party’s training data. This notion of privacy can be formalized using the language of *differential privacy* [66]. Intuitively, differential privacy allows a learning algorithm to capture patterns that hold for large groups of users, while withholding information that is unique to any individual user. More precisely, a (randomized) learning algorithm is differentially

private if it outputs a particular model with roughly the same probability whether a particular user’s data was in the training set or not:

Definition 5.7 (Differential Privacy). An algorithm `func` satisfies (ϵ, δ) -differential privacy [65], if for any datasets D, D' that differ in one record, and any set of outputs S :

$$\Pr[\text{func}(D) \in S] \leq e^\epsilon \Pr[\text{func}(D') \in S] + \delta.$$

The parameter ϵ is referred to as the *privacy budget*. Informally, the lower the value of ϵ , the more private the algorithm is. Setting the value of the privacy budget is an intricate problem, and common wisdom suggests setting the budget to some small value such as $\ln(2)$ or $\ln(3)$ [66]. For deep learning tasks, the value of the (provable upper-bound on the) privacy budget has typically been set to larger values such as $\epsilon = 3$ [198] or $\epsilon = 8$ [1]. The additive error δ should be chosen to be lower than $1/|D|$.⁷ Typical values used in the literature include $1/2|D|$ and $1/|D|^{1.1}$.

The first algorithms for differentially private Empirical Risk Minimization [34] relied on a theoretical analysis that does not appear amenable to the training of deep neural networks. Later works proposed privacy-preserving versions of stochastic gradient descent (SGD) [11, 239], which were then adapted to the special case of training deep neural networks [1, 234]. In particular, the seminal work of Abadi et al. [1] introduced, analyzed and evaluated DP-SGD, an algorithm tailored to deep neural networks that achieved promising tradeoffs between a trained model’s accuracy and the privacy budget ϵ .

Achieving privacy without sacrificing utility. In Chapter 6 and Chapter 7, we will consider, respectively, the tasks of training machine learning models with differential privacy guarantees, and of deploying a private machine learning prediction service using secure computation. For both tasks, we introduce new techniques to obtain privacy-utility tradeoffs that are orders-of-magnitude better than in prior work.

In Chapter 6, we introduce new techniques for differentially private training of neural networks. Prior work found that state-of-the-art deep neural networks were hard to train with strong differential privacy guarantees, and instead proposed specific network architectures tailored for DP-SGD [1, 197, 198]. We take this idea a step further, and ask whether we could obtain better results by foregoing deep learning entirely! We find that standard *feature engineering* techniques—which are routinely outperformed by end-to-end deep learning in the non-private setting—can lead to state-of-the-art differentially private models for a variety of canonical vision tasks. We further show that when given access to a large *public* dataset for a non-sensitive but related task, we can transfer features learned on the public dataset to train differentially private models with close to no performance overhead.

⁷An $(0, 1/|D|)$ -differentially private algorithm may be blatantly non private and output a randomly chosen record in D with probability 1.

In Chapter 7, we introduce Slalom, a system for securely outsourcing neural network predictions to a remote cloud provider equipped with a TEE. While TEEs offer a seemingly simple solution to this problem, they still incur a large performance overhead compared to the custom hardware that is typically used to run modern neural networks (e.g., a GPU or a TPU [125]). The approach taken in Slalom is thus to further *outsource* part of the work performed by the TEE to these faster (but untrusted) processors. Our protocol splits up a neural network evaluation into computationally expensive matrix multiplications—which we show can be outsourced without compromising privacy or integrity—and inexpensive non-linear activations that the TEE computes itself. Compared to a baseline that runs all computations in the TEE, Slalom increases throughput and energy efficiency by one order of magnitude.

The work in the second part of this dissertation demonstrates new avenues to achieve strong privacy guarantees for machine learning, at a much lower performance cost than in prior work.

Chapter 6

Differentially Private Learning With Better Features

Training deep neural networks with strong differential privacy (DP) guarantees comes at a significant cost in utility [1, 10, 76, 284]. In fact, on many ML benchmarks the reported accuracy of private deep learning still falls short of “shallow” (non-private) techniques. For example, on CIFAR-10, Papernot et al. [198] train a neural network to 66.2% accuracy for a large DP budget of $\epsilon = 7.53$, the highest accuracy we are aware of for this privacy budget. Yet, without privacy, higher accuracy is achievable with linear models and non-learned “handcrafted” features, e.g., [52, 187]. This leads to the central question of this chapter:

Can differentially private learning benefit from handcrafted features?

We answer this question affirmatively by introducing simple and strong handcrafted baselines for differentially private learning, that significantly improve the privacy-utility guarantees on canonical vision benchmarks.

We leverage the *Scattering Network* (ScatterNet) of Oyallon and Mallat [187]—a *non-learned* SIFT-like feature extractor [155]—to train linear models that improve upon the privacy-utility guarantees of deep learning on MNIST, Fashion-MNIST and CIFAR-10 (see Table 6.1). For example, on CIFAR-10 we exceed the accuracy reported by Papernot et al. [198] while simultaneously improving the provable DP-guarantee by $130\times$. On MNIST, we match the privacy-utility guarantees obtained with PATE [196] *without requiring access to any public data*. We find that privately training deeper neural networks on handcrafted features also significantly improves over end-to-end deep learning, and even slightly exceeds the simpler linear models on CIFAR-10. Our results show that private deep learning remains outperformed by handcrafted priors on many tasks, and thus has yet to reach its “*AlexNet moment*” [142].

Table 6.1: **Test accuracy of models with handcrafted ScatterNet features compared to prior results with end-to-end CNNs for various DP budgets** ($\epsilon, \delta = 10^{-5}$). Lower ϵ values provide stronger privacy. The end-to-end CNNs with maximal accuracy for each privacy budget are underlined. We select the best ScatterNet model for each DP budget $\epsilon \leq 3$ with a hyper-parameter search, and show the mean and standard deviation in accuracy for five runs.

Data	ϵ -DP	Source	Test Accuracy (%)		
			CNN	ScatterNet+linear	ScatterNet+CNN
MNIST	1.2	Feldman and Zrnic [77]	<u>96.6</u>	98.1 \pm 0.1	97.8 \pm 0.1
	2.0	Abadi et al. [1]	95.0	98.5 \pm 0.0	98.4 \pm 0.1
	2.32	Bu et al. [23]	96.6	98.6 \pm 0.0	98.5 \pm 0.0
	2.5	Chen and Lee [35]	90.0	98.7 \pm 0.0	98.6 \pm 0.0
	2.93	Papernot et al. [197]	<u>98.1</u>	98.7 \pm 0.0	98.7 \pm 0.1
	3.2	Nasr et al. [179]	96.1	–	–
	6.78	Yu et al. [283]	93.2	–	–
Fashion-MNIST	2.7	Papernot et al. [197]	<u>86.1</u>	89.5 \pm 0.0	88.7 \pm 0.1
	3.0	Chen and Lee [35]	82.3	89.7 \pm 0.0	89.0 \pm 0.1
CIFAR-10	3.0	Nasr et al. [179]	<u>55.0</u>	67.0 \pm 0.1	69.3 \pm 0.2
	6.78	Yu et al. [283]	44.3	–	–
	7.53	Papernot et al. [197]	<u>66.2</u>	–	–
	8.0	Chen and Lee [35]	53.0	–	–

We find that models with handcrafted features outperform end-to-end deep models, *despite having more trainable parameters*. This is counter-intuitive, as the guarantees of private learning degrade with dimensionality in the worst case [11].¹ We explain the benefits of handcrafted features by analyzing the convergence rate of *non-private* gradient descent. First, we observe that with low enough learning rates, training converges similarly with or without privacy (both for models with and without handcrafted features). Second, we show that handcrafted features significantly boost the convergence rate of *non-private* learning at low learning rates. As a result, when training with privacy, handcrafted features lead to more accurate models for a fixed privacy budget.

Considering these results, we ask: *what is the cost of private learning’s “AlexNet moment”?* That is, which additional resources do we need in order to outperform our private handcrafted baselines? Following McMahan et al. [166], we first consider the *data complexity* of private end-to-end learning. On CIFAR-10, we use an additional 500,000 labeled Tiny Images from Carmon et al. [33] to show that about *an order of magnitude* more private training data is needed for end-to-end deep models to outperform our handcrafted features baselines. The high sample-complexity of private deep learning could be detrimental for tasks that cannot leverage “internet-scale” data collection (e.g., most medical applications).

We further consider private learning with access to *public* data from a similar domain. In this

¹A number of recent works have attempted to circumvent this worst-case dimensionality dependence by leveraging the empirical observation that model gradients lie in a low-dimensional subspace [127, 289].

setting, handcrafted features can be replaced by features learned from public data via *transfer learning* [210]. While differentially private transfer learning has been studied in prior work [1, 197], we find that its privacy-utility guarantees have been underestimated. We revisit these results and show that with transfer learning, strong privacy comes at only a minor cost in accuracy. For example, given public *unlabeled* ImageNet data, we train a CIFAR-10 model to 92.7% accuracy for a DP budget of $\epsilon = 2$.

Our results demonstrate that higher quality features—whether handcrafted or transferred from public data—are of paramount importance for improving the performance of private classifiers in low (private) data regimes.

6.1 Preliminaries

We consider the standard *central* model of differential privacy (DP): a trusted party trains an ML model f on a private dataset D , and publicly releases the model. The learning algorithm A satisfies (ϵ, δ) -differential privacy [65], if for any datasets D, D' that differ in one record, and any set of models S :

$$\Pr[A(D) \in S] \leq e^\epsilon \Pr[A(D') \in S] + \delta.$$

DP bounds an adversary’s ability to infer information about any individual training point from the model. Cryptography can split the trust in a central party across users [19, 123].

Prior work has trained private deep neural networks “end-to-end” (e.g., from image pixels), with large losses in utility [1, 198, 234]. In contrast, we study the benefits of handcrafted features that encode *priors* on the learning task’s public domain (e.g., edge detectors for images). Although end-to-end neural networks outperform such features in the non-private setting, our thesis is that handcrafted features result in an easier learning task that is more amenable to privacy. We focus on computer vision, a canonical domain for private deep learning [1, 179, 198, 283]), with a rich literature on handcrafted features [22, 58, 155]. Our approach can be extended to handcrafted features in other domains, e.g., text or speech.

6.1.1 Scattering Networks

We use the Scattering Network (ScatterNet) of Oyallon and Mallat [187], a feature extractor that encodes natural image priors (e.g., invariance to small rotations and translations) using a cascade of wavelet transforms [22]. As this cascade of transforms is data independent, we can obtain a differentially private classifier by privately fine-tuning a (linear) model on top of locally extracted features.

Given an input x , the output of a scattering network of depth J is a feature vector given by

$$S(x) := A_J |W_2 |W_1 x| |, \tag{6.1}$$

where the operators W_1 and W_2 are complex-valued wavelet transforms, each followed by a non-linear complex modulus, and the final operator A performs spatial averaging over patches of 2^J features. Both wavelet transforms W_1 and W_2 are linear operators that compute a cascade of convolutions with filters from a fixed family of wavelets. For an input image of spatial dimensions $H \times W$, the ScatterNet is applied to each of the image’s color channels independently to yield an output tensor of dimension $(K, \frac{H}{2^J}, \frac{W}{2^J})$. The channel dimensionality K depends on the network depth J and the granularity of the wavelet filters, and is chosen so that $K/2^{2J} = O(1)$ (i.e., the ScatterNet approximately preserves the data dimensionality).

For all experiments, we use the default parameters proposed by Oyallon and Mallat [187], namely a Scattering Network of depth $J = 2$, consisting of wavelet filters rotated along eight angles. For an input image of spatial dimensions $H \times W$, this configuration produces an output of dimension $(K, H/4, W/4)$, with $K = 81$ for grayscale images, and $K = 243$ for RGB images. Note that the transform is thus *expansive*.

Why ScatterNets? In this paper, we propose to use the ScatterNet features of Oyallon and Mallat [187] as a basis for shallow differentially private vision classifiers. We briefly discuss a number of other shallow approaches that produce competitive results for canonical vision tasks, but which appear less suitable for private learning.

Unsupervised feature dictionaries. Coates and Ng [52] achieve above 80% test accuracy on CIFAR-10 with linear models trained on top of a dictionary of features extracted from a mixture of image patches. Their approach relies on a combination of many ‘tricks’, including data normalization, data whitening, tweaks to standard Gaussian-Mixture-Model (GMM) algorithms, feature selection, etc. While it is conceivable that each of these steps could be made differentially private, we opt here for a much simpler unlearned baseline that is easier to analyze and to apply to a variety of different tasks. We note that existing work on differentially-private learning of mixtures (e.g., [181]) has mainly focused on asymptotic guarantees, and we are not aware of any exiting algorithms that have been evaluated on high-dimensional datasets such as CIFAR-10.

Kernel Machines. Recent work on *Neural Tangent Kernels* [121] has shown that the performance of deep neural networks on CIFAR-10 could be matched by specialized kernel methods [5, 150, 227]. Unfortunately, private learning with non-linear kernels is intractable in general [34, 216]. Chaudhuri et al. [34] propose to obtain private classifiers by approximating kernels using *random features* [208], but the very high dimensionality of the resulting learning problem makes it challenging to outperform our handcrafted features baseline. Indeed, we had originally considered a differentially-private variant of the random-feature CIFAR-10 classifier proposed in [211], but found the model’s high dimensionality (over 10 million features) to be detrimental to private learning.

input : Data $\{x^{(1)} \dots, x^{(N)}\}$, learning rate η , noise scale σ , batch size B , gradient norm bound C , epochs T

- 1 Initialize θ_0 randomly
- for** $t \in [T \cdot N/B]$ **do**
 - 2 Sample a batch \mathbf{B}_t by selecting each $x^{(i)}$ independently with probability B/N
 - 3 For each $x \in \mathbf{B}_t$: $g_t(x) \leftarrow \nabla_{\theta_t} L(\theta_t, x^{(i)})$ // compute per-sample gradients
 - 4 $\tilde{g}_t(x) \leftarrow g_t(x) \cdot \min(1, C/\|g_t(x)\|_2)$ // clip gradients
 - 5 $\tilde{g}_t \leftarrow \frac{1}{B} (\sum_{x \in \mathbf{B}_t} \tilde{g}_t(x) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$ // add Gaussian noise to average gradient
 - 6 $\theta_{t+1} \leftarrow \theta_t - \eta \tilde{g}_t$ // SGD step

end
output: $\theta_{TN/B}$

Algorithm 3: The DP-SGD Algorithm [1].

6.1.2 Differentially Private Stochastic Gradient Descent

Throughout this work, we use the DP-SGD algorithm of Abadi et al. [1] (see Algorithm 3).

The tightest known privacy analysis of the DP-SGD algorithm is based on the notion of Rényi differential privacy (RDP) from Mironov [170], which we recall next.

Definition 6.1 (Rényi Divergence). For two probability distributions \mathcal{P} and \mathcal{Q} defined over a range \mathbb{Z} , the Rényi divergence of order $\alpha > 1$ is

$$D_\alpha(\mathcal{P} \parallel \mathcal{Q}) := \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim \mathcal{Q}} \left(\frac{\mathcal{P}(x)}{\mathcal{Q}(x)} \right)^\alpha.$$

Definition 6.2 ((α, ε) -RDP [170]). A randomized mechanism $A : \mathbb{D} \rightarrow \mathbb{Z}$ is said to have ε -Rényi differential privacy of order α , or (α, ε) -RDP for short, if for any adjacent $D, D' \in \mathbb{D}$ it holds that

$$D_\alpha(A(D) \parallel A(D')) \leq \varepsilon.$$

To analyze the privacy guarantees of DP-SGD, we numerically compute $D_\alpha(A(D) \parallel A(D'))$ for a range of orders α [171, 269] in each training step, where D and D' are training sets that differ in a single element. To obtain privacy guarantees for t training steps, we use the composition properties of RDP:

Lemma 6.3 (Adaptive composition of RDP [171]). *Let $A : \mathbb{D} \rightarrow \mathcal{R}_1$ be (α, ε_1) -RDP and $A' : \mathbb{Z}_1 \times \mathbb{D} \rightarrow \mathbb{Z}_2$ be (α, ε_2) -RDP, then the mechanism defined as (X, Y) , where $X \sim A(D)$ and $Y \sim A'(X, D)$, satisfies $(\alpha, \varepsilon_1 + \varepsilon_2)$ -RDP.*

Finally, the RDP guarantees of the full DP-SGD procedure can be converted into a (ε, δ) -DP guarantee:

Lemma 6.4 (From RDP to (ϵ, δ) -DP [171]). *If A is an (α, ϵ) -RDP mechanism, it also satisfies $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$ -DP for any $0 < \delta < 1$.*

6.1.3 Differentially Private ScatterNet Classifiers

To train private classifiers, we use the DP-SGD algorithm² of Abadi et al. [1] (see Section 6.1.2). DP-SGD works as follows: (1) batches of expected size B are sampled at random;³ (2) gradients are *clipped* to norm C ; (3) Gaussian noise of variance $\sigma^2 C^2 / B^2$ is added to the mean gradient. DP-SGD guarantees privacy for *gradients*, and is thus oblivious to preprocessing applied independently to each data sample, such as the ScatterNet transform.

When training a supervised classifier on top of ScatterNet features with gradient descent, we find that *normalizing* the features is crucial to obtain strong performance. We consider two approaches:

- *Group Normalization* [273]: the channels of $S(x)$ are split into G groups, and each is normalized to zero mean and unit variance. Data points are normalized independently so this step incurs no privacy cost.
- *Data Normalization*: the channels of $S(x)$ are normalized by their mean and variance across the training data. This step incurs a privacy cost as the per-channel means and variances need to be privately estimated.

Evaluation of feature normalization. To evaluate the effect of feature normalization in Table 6.2, we train linear models on ScatterNet features using DP-SGD without noise ($\sigma = 0$). We train one model without feature normalization, one with Data Normalization, and three with Group Normalization [273] with $G \in \{9, 27, 81\}$ groups. The hyper-parameters are given below. For each dataset, we report the best choice for the group size G .

Parameter	MNIST	Fashion-MNIST	CIFAR-10
Gradient clipping norm C	0.1	0.1	0.1
Momentum	0.9	0.9	0.9
Epochs T	20	20	20
Batch size B	512	512	512
Learning rate η	2	4	2
Best choice of groups G	27	81	27

²Yu et al. [282] show that DP-SGD outperforms other algorithms for private convex optimization, e.g., logistic regression with output or objective perturbation [11, 34, 133]. In Section 6.5.2, we show that DP-SGD also outperforms *Privacy Amplification by Iteration* [78] in our setting.

³Existing DP-SGD implementations such as `TensorFlow/privacy` (<https://github.com/tensorflow/privacy>) and `Opacus` (<https://github.com/pytorch/opacus>), as well as many prior works (e.g., [1, 198]) heuristically split the data into random batches of size *exactly* B . We use the same heuristic and show in Section 6.5.3 that using the correct batch sampling does not affect our results.

Table 6.2: **Effect of feature normalization on the test accuracy of *non-private* ScatterNet models after 20 epochs.** We also report the maximal test accuracy upon convergence (mean and standard deviation over five runs).

Dataset	Normalization (Test accuracy after 20 epochs)			Maximal Accuracy
	None	Group Normalization	Data Normalization	
MNIST	95.9 ± 0.0	99.1 ± 0.0	99.1 ± 0.0	99.3 ± 0.0
Fashion-MNIST	82.6 ± 0.1	90.9 ± 0.1	91.0 ± 0.2	91.5 ± 0.0
CIFAR-10	58.0 ± 0.1	67.8 ± 0.2	70.7 ± 0.1	71.1 ± 0.0

Results for training linear ScatterNet models with various feature normalization techniques are in Table 6.2. Normalization significantly accelerates convergence of *non-private* linear models trained on ScatterNet features, for MNIST, Fashion-MNIST and CIFAR-10. For CIFAR-10, Data Normalization performs significantly better than Group Normalization, so the small privacy cost of estimating channel statistics is warranted. While the maximal test accuracy of these models falls short of state-of-the-art CNNs, it exceeds all previously reported results for differentially private neural networks (even for large privacy budgets).

Private feature normalization. To privately apply *Data Normalization* to the ScatterNet features (which greatly improves convergence, especially on CIFAR-10), we use the `PrivDataNorm` procedure in Algorithm 4 to compute private estimates of the per-channel mean and variance of the ScatterNet features.

In order to obtain tight privacy guarantees for the full training procedure (i.e., privacy-preserving Data Normalization followed by DP-SGD), we first derive the RDP guarantees of `PrivDataNorm`:

Claim 6.5. *The `PrivDataNorm` procedure is $(\alpha, \alpha/\sigma_{norm}^2)$ -RDP for any $\alpha > 1$.*

The above claim follows from the RDP guarantees of the Gaussian mechanism in [170], together with the composition properties of RDP in Lemma 6.3 above.

Finally, given an RDP guarantee of (α, ε_1) for `PrivDataNorm`, and an RDP guarantee of (α, ε_2) for DP-SGD, we apply Lemma 6.3 to obtain an RDP guarantee of $(\alpha, \varepsilon_1 + \varepsilon_2)$, and convert to a DP guarantee using Lemma 6.4.

6.2 Evaluating Private ScatterNet Classifiers

We compare differentially private ScatterNet classifiers and deep learning models on MNIST [145], Fashion-MNIST [274] and CIFAR-10 [141]. Many prior works report improvements over the DP-SGD procedure of Abadi et al. [1] for these datasets. As we will show, ScatterNet classifiers outperform all

```

Function PrivChannelMean(data  $D \in \mathbb{R}^{N \times K \times H \times W}$ , norm bound  $C$ , noise scale  $\sigma_{norm}$ )
1 | For  $1 \leq i \leq N$ :  $\mu_i \leftarrow \mathbb{E}_{h,w} [D_{(i, \cdot, h, w)}] \in \mathbb{R}^K$  // compute per-channel means
2 |  $\mu_i \leftarrow \mu_i \cdot \min(1, C/\|\mu_i\|_2)$  // clip each sample's per-channel means
3 |  $\tilde{\mu} \leftarrow \mathbb{E}_i[\mu_i] + \frac{1}{N}\mathcal{N}(0, \sigma_{norm}^2 C^2 \mathbf{I})$  // private mean using Gaussian mechanism
4 | return  $\tilde{\mu}$ 

Function PrivDataNorm(data  $D$ , norm bounds  $C_1, C_2$ , noise scale  $\sigma_{norm}$ , threshold  $\tau$ )
1 |  $\tilde{\mu} \leftarrow \text{PrivChannelMean}(D, C_1, \sigma_{norm})$  // private per-channel mean
2 |  $\tilde{\mu}_{D^2} \leftarrow \text{PrivChannelMean}(D^2, C_2, \sigma_{norm})$  // private per-channel mean-square
3 |  $\tilde{Var} \leftarrow \max(\tilde{\mu}_{D^2} - \tilde{\mu}^2, \tau)$  // private per-channel variance
4 | For each  $1 \leq i \leq N$ ,  $\hat{D}_i \leftarrow (D_i - \tilde{\mu})/\sqrt{\tilde{Var}}$  // normalize each sample
5 | return  $\hat{D}$ 

```

Algorithm 4: Private Data Normalization.

prior approaches *while making no algorithmic changes to DP-SGD*. ScatterNet classifiers can thus serve as a strong baseline for evaluating proposed improvements over DP-SGD in the future.

6.2.1 Experimental Setup

Most prior works find the best model for a given DP budget using a hyper-parameter search. As the private training data is re-used many times, this overestimates the privacy guarantees. Private hyper-parameter search is possible at a small cost in the DP budget [152], but we argue that fully accounting for this privacy leakage is hard as even our choices of architectures, optimizers, hyper-parameter *ranges*, etc. are informed by prior analysis of the same data. As in prior work, we thus do not account for this privacy leakage, and instead compare ScatterNet models and end-to-end CNNs with similar hyper-parameter searches. Moreover, we find that ScatterNet models are very robust to hyper-parameter changes and achieve near-optimal utility with random hyper-parameters (see Table 6.7). To evaluate ScatterNet models, we apply the following hyper-parameter search:

- We begin by fixing a *privacy schedule*. We target a moderate differential privacy budget of ($\varepsilon = 3, \delta = 10^{-5}$) and compute the noise scale σ of DP-SGD so that the privacy budget is consumed after T epochs. We try different values of T , with larger values resulting in training for more steps but with higher noise.
- We fix the gradient clipping threshold for DP-SGD to $C = 0.1$ for all our experiments. Thakkar et al. [247] suggest to vary this threshold adaptively, but we did not observe better performance by doing so.
- We try various batch sizes B and base learning rates η , with linear learning rate scaling [97].⁴

⁴Our decision to try various batch sizes is inspired by Abadi et al. [1] who found that this parameter has a large

Table 6.3: **Hyper-parameters for the evaluation of differentially private classifiers.** We use the same set of parameters for private linear classifiers fine-tuned on ScatterNet features, CNNs fine-tuned on ScatterNet features, and end-to-end CNNs.

Parameter	MNIST	Fashion-MNIST	CIFAR-10
DP guarantee (ϵ, δ)	$(3, 10^{-5})$	$(3, 10^{-5})$	$(3, 10^{-5})$
Gradient clipping norm C	0.1	0.1	0.1
Momentum	0.9	0.9	0.9
Batch size B	$\{512, 1024, \dots, 16384\}$	$\{512, 1024, \dots, 16384\}$	$\{512, 1024, \dots, 16384\}$
Learning rate η	$\{1/4, 1/2, 1, 2\} \cdot B/512$	$\{1/4, 1/2, 1, 2\} \cdot B/512$	$\{1/8, 1/4, 1/2, 1\} \cdot B/512$
Epochs T	$\{15, 25, 40\}$	$\{15, 25, 40\}$	$\{30, 60, 120\}$
DP-SGD noise scale σ	calculated numerically so that (ϵ, δ) -DP is spent after T epochs		
Group Norm. groups G	$\{9, 27, 81\}$	$\{9, 27, 81\}$	$\{9, 27, 81\}$
Data Norm. $(C_1, C_2, \sigma_{\text{norm}})$	$(0.2, 0.05, \{6, 8\})$	$(0.3, 0.15, \{6, 8\})$	$(1.0, 1.5, \{6, 8\})$

- We try both Group Normalization [273] with different choices for the number of groups, and private Data Normalization with different choices of privacy budgets (see Section 6.1.2 for details).

We perform a grid-search over all parameters as detailed in Table 6.3. We compare our ScatterNet classifiers to the CNN models of Papernot et al. [198], which achieve the highest reported accuracy for our targeted privacy budget for all three datasets. We also perform a grid-search for these models, which reproduces the results of Papernot et al. [198]. We use the ScatterNet implementation from `Kymatio` [4], and the DP-SGD implementation in `opacus`⁵ (formerly called `pytorch-dp`).

We use DP-SGD with momentum for all experiments. Prior work found that the use of adaptive optimizers (e.g., Adam [134]) provided only marginal benefits for private learning [197]. Moreover, we use no data augmentation, weight decay, or other mechanisms aimed at preventing overfitting. The reason is that differential privacy is itself a powerful regularizer (informally, differential privacy implies low generalization error [67]), so our models all *underfit* the training data.

We use a NVIDIA Titan Xp GPU with 12GB of RAM for all our experiments. To run DP-SGD with large batch sizes B , we use the “virtual batch” approach of `opacus`: the average of clipped gradients is accumulated over multiple “mini-batches”; once B gradients have been averaged, we add noise and take a gradient update step.

effect on the performance of DP-SGD. Yet, in Section 6.5.1 we show empirically, and argue formally that with a *linear learning rate scaling* [97], DP-SGD performs similarly for a range of batch sizes. As a result, we recommend following the standard approach for tuning non-private SGD, wherein we fix the batch size and tune the learning rate.

⁵<https://github.com/pytorch/opacus>. Accessed 2021-6-22.

6.2.2 Model Architectures

Linear ScatterNet classifiers. The default Scattering Network of Oyallon and Mallat [187] extracts feature vectors of size $(81, 7, 7)$ for MNIST and Fashion-MNIST and of size $(243, 8, 8)$ for CIFAR-10. We then train a standard logistic regression classifier (with per-class bias) on top of these features, as summarized below:

Dataset	Image size	Linear ScatterNet size
MNIST	28×28	3969×10
Fashion-MNIST	28×28	3969×10
CIFAR-10	$32 \times 32 \times 3$	15552×10

End-to-end CNNs. We use the CNN architectures proposed by Papernot et al. [198], which were found as a result of an architecture search tailored to DP-SGD.⁶ Notably, these CNNs are quite small (since the noise of DP-SGD grows with the model’s dimensionality) and use Tanh activations, which Papernot et al. [198] found to outperform the more common ReLU activations. For the experiments in Section 6.3, we also consider a smaller CIFAR-10 model, with a dimensionality comparable to the linear ScatterNet classifier. While the standard model has six convolutional layers of size 32-32-64-64-128-128, the smaller model has five convolutional layers of size 16-16-32-32-64 (with max-pooling after the 2nd, 4th and 5th convolution).

Table 6.4: **Architecture of end-to-end CNN models.** The models are from [198] and use Tanh activations. In Section 6.3, we also use a smaller variant of the CIFAR-10 model with five convolutional layers of 16-16-32-32-64 filters.

(a) MNIST and Fashion-MNIST.		(b) CIFAR-10.	
Layer	Parameters	Layer	Parameters
Convolution	16 filters of 8x8, stride 2, padding 2	Convolution x2	32 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 1	Max-Pooling	2x2, stride 2
Convolution	32 filters of 4x4, stride 2, padding 0	Convolution x2	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 1	Max-Pooling	2x2, stride 2
Fully connected	32 units	Convolution x2	128 filters of 3x3, stride 1, padding 1
Fully connected	10 units	Max-Pooling	2x2, stride 2
		Fully connected	128 units
		Fully connected	10 units

ScatterNet CNNs. To fine-tune CNNs on top of ScatterNet features, we adapt the CNNs from Table 6.4. As the ScatterNet feature vector is larger than the input image ($784 \rightarrow 3969$ features for MNIST and Fashion-MNIST, and $3072 \rightarrow 15552$ features for CIFAR-10), we use smaller CNN

⁶The CNN architecture for CIFAR-10 in Table 6.4 differs slightly from that described in [198]. Based on discussions with the authors of [198], the architecture in Table 6.4 is the correct one to reproduce their best results.

models. For MNIST and Fashion MNIST, we reduce the number of convolutional filters. For CIFAR-10, we reduce the network depth from 8 to 3, which results in a model with approximately as many parameters as the linear ScatterNet classifier.

Table 6.5: **Architecture of CNN models fine-tuned on ScatterNet features.** The models use Tanh activation.

(a) MNIST and Fashion-MNIST.		(b) CIFAR-10.	
Layer	Parameters	Layer	Parameters
Convolution	16 filters of 3x3, stride 2, padding 1	Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 1	Max-Pooling	2x2, stride 2
Convolution	32 filters of 3x3, stride 1, padding 1	Convolution	64 filters of 3x3, stride 1, padding 1
Max-Pooling	2x2, stride 1	Max-Pooling	2x2, stride 2
Fully connected	32 units	Fully connected	10 units
Fully connected	10 units		

Non-private accuracy. For each of the model architectures described above, we report the best achieved test accuracy without privacy, and without any other form of explicit regularization. For MNIST and Fashion-MNIST, fine-tuning a linear model or a CNN on top of ScatterNet features results in similar performance, whereas on CIFAR-10, the CNN performs slightly better. For Fashion-MNIST the end-to-end CNN performs slightly worse than the linear model (mainly due to a lack of regularization). For CIFAR-10, the end-to-end CNN significantly outperforms the ScatterNet models.

Table 6.6: **Test accuracy for models trained without privacy.** Average and standard deviation are computed over five runs.

Dataset	ScatterNet+Linear	ScatterNet+CNN	CNN
MNIST	99.3 \pm 0.0	99.2 \pm 0.0	99.2 \pm 0.0
Fashion-MNIST	91.5 \pm 0.0	91.5 \pm 0.2	90.1 \pm 0.2
CIFAR-10	71.1 \pm 0.0	73.8 \pm 0.3	80.0 \pm 0.1

6.2.3 Results

To measure a classifier’s accuracy for a range of privacy budgets, we compute the test accuracy as well as the DP budget ϵ after each training epoch (with the last epoch corresponding to $\epsilon = 3$). For various DP budgets ($\epsilon, \delta = 10^{-5}$) used in prior work, Table 6.1 shows the maximal test accuracy achieved by a linear ScatterNet model in our hyper-parameter search, averaged over five runs. We also report results with CNNs trained on ScatterNet models, which are described in more detail below. Figure 6.1 further compares the full privacy-accuracy curves of our linear ScatterNets and of the CNNs of Papernot et al. [198]. Linear models with handcrafted features significantly outperform

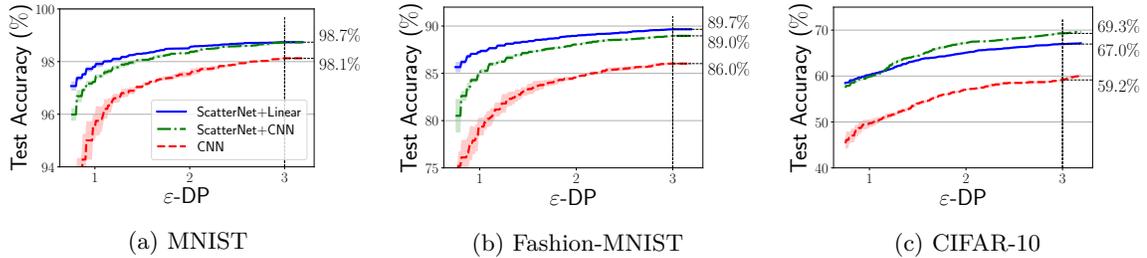


Figure 6.1: **Privacy-accuracy tradeoffs for ScatterNet classifiers.** Shows the highest test accuracy achieved for each DP budget ($\epsilon, \delta = 10^{-5}$) for linear ScatterNet classifiers, CNNs on top of ScatterNet features, and end-to-end CNNs. Shows mean and standard deviation across five runs.

prior results with end-to-end CNNs, for all privacy budgets $\epsilon \leq 3$ we consider. Even when prior work reports results for larger budgets, they do not exceed the accuracy of our baseline.

In particular, for CIFAR-10, we match the best CNN accuracy in [198]—namely 66.2% for a budget of $\epsilon = 7.53$ —with a much smaller budget of $\epsilon = 2.6$. This is an improvement in the DP-guarantee of $e^{4.9} \approx 134$. On MNIST, we significantly improve upon CNN models, and match the results of PATE [196], namely 98.5% accuracy at $\epsilon = 1.97$, in a more restricted setting (PATE uses 5,000 *public* unlabeled MNIST digits).

Training CNNs on handcrafted features. Since *linear* models trained on handcrafted features outperform deep models trained end-to-end, a natural question is whether training deeper models on these features achieves even better results. We repeat the above experiment with a similar CNN model trained on ScatterNet features (see Table 6.5). The privacy-accuracy curves for these models are in Figure 6.1. We find that handcrafted features also improve the utility of private deep models, a phenomenon which we analyze and explain in Section 6.3. On CIFAR-10, the deeper ScatterNet models even slightly outperform the linear models, while for MNIST and Fashion-MNIST the linear models perform best. This can be explained by the fact that in the *non-private setting*, linear ScatterNet models achieve close to state-of-the-art accuracy on MNIST and Fashion-MNIST, and thus there is little room for improvement with deeper models (see Table 6.6). Table 6.7 further shows that ScatterNet CNNs are also less sensitive to hyper-parameters than end-to-end CNNs.

Note that on each dataset we consider, end-to-end CNNs can outperform ScatterNet models when trained *without privacy*. Thus, end-to-end CNNs trained with DP-SGD must eventually surpass ScatterNet models for large enough privacy budgets. But this currently requires settling for weak provable privacy guarantees. On CIFAR-10 for example, ScatterNet classifiers still outperform end-to-end CNNs for $\epsilon = 7.53$ [198]. While the analysis of DP-SGD might not be tight, Jagielski et al. [122] suggest that the true ϵ guarantee of DP-SGD is at most one order of magnitude smaller than the current analysis suggests. Thus, surpassing handcrafted features for small privacy budgets on CIFAR-10 may require improvements beyond a tighter analysis of DP-SGD.

Table 6.7: **Accuracy variability across hyper-parameters.** For each model, we report the minimum, maximum, median and median absolute deviation (MAD) in test accuracy (in %) achieved for a DP budget of $(\epsilon = 3, \delta = 10^{-5})$. The maximum accuracy below may exceed those in Table 6.1 and Figure 6.1, which are averages of five runs. SN stands for ScatterNet.

Model	MNIST				Fashion-MNIST				CIFAR-10			
	Min	Max	Median	MAD	Min	Max	Median	MAD	Min	Max	Median	MAD
SN + Linear	96.8	98.8	98.4	0.2	85.3	89.8	88.7	0.5	59.5	67.0	65.4	0.9
SN + CNN	95.6	98.8	98.1	0.3	77.8	89.1	87.2	1.0	57.3	69.5	66.9	1.6
CNN	86.1	98.2	97.4	0.5	20.2	86.2	83.6	1.8	39.4	59.2	52.5	5.4

6.2.4 Analysis of Hyper-parameters

As noted in Section 6.2.1, our models (and those of most prior work) are the result of a hyper-parameter search. While we do not account for the privacy cost of this search, Table 6.7 shows that an additional advantage of ScatterNet classifiers is an increased robustness to hyper-parameter changes. In particular, for CIFAR-10 the *worst* configuration for linear ScatterNet classifiers outperforms the *best* configuration for end-to-end CNNs. Moreover, on MNIST and Fashion-MNIST, the median accuracy of linear ScatterNet models outperforms the best end-to-end CNN.

In the table below, we provide the hyper-parameters that result in the highest test accuracy for our target DP budget of $(\epsilon = 3, \delta = 10^{-5})$. We did not consider larger privacy budgets for ScatterNet classifiers, as the accuracy we achieve at $\epsilon = 3$ is close to the accuracy of non-private ScatterNet models (see Table 6.2). For each model, we report the *base* learning rate, before re-scaling by $B/512$.

Parameter	MNIST			Fashion-MNIST			CIFAR-10		
	SN+Linear	SN+CNN	CNN	SN+Linear	SN+CNN	CNN	SN+Linear	SN+CNN	CNN
Batch size B	4096	1024	512	8192	2048	2048	8192	8192	1024
Base LR η	1	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$
Epochs T	40	25	40	40	40	40	60	60	30
Groups G	-	-	-	27	27	-	-	-	-
Data Norm. σ_{norm}	8	8	-	-	-	-	8	8	-

We find that some hyper-parameters that result in the best performance are at the boundary of our search range. Yet, as we show in Figure 6.2, modifying these hyper-parameters results in no significant upward trend, so we refrained from further increasing our search space.

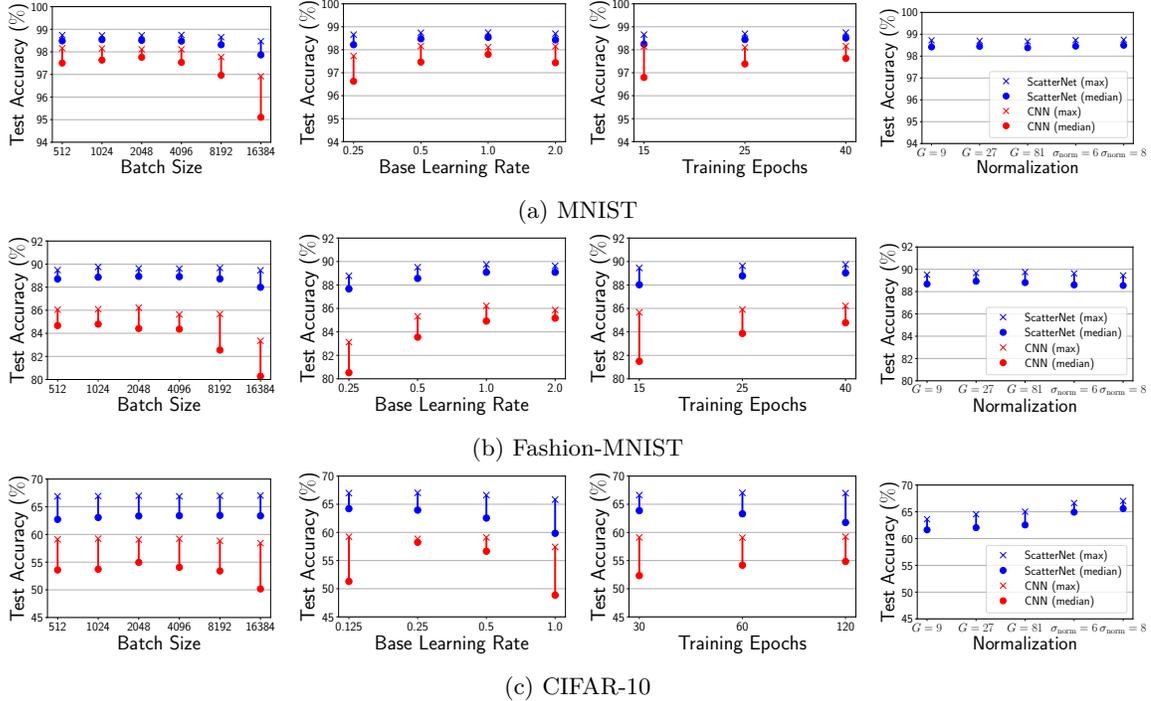


Figure 6.2: **Median and maximum test accuracy with one hyper-parameter fixed.** Shows the median and maximum test accuracy of linear ScatterNet classifiers and end-to-end CNNs when we fix one hyper-parameter in Table 6.3 and run a grid-search over all others (for a privacy budget of $(\epsilon = 3, \delta = 10^{-5})$).

Figure 6.2 shows the median and maximum model performance for different choices of a single parameter. The median and maximum are computed over all choices for the other hyper-parameters in Table 6.3. As we can see, the maximal achievable test accuracy is remarkably stable when fixing one of the algorithm’s hyper-parameters, with the exception of overly large batch sizes or overly low learning rates for end-to-end CNNs.

6.3 How Do Handcrafted Features Help?

In this section, we analyze *why* private models with handcrafted features outperform end-to-end CNNs. We first consider the *dimensionality* of our models, but show that this does not explain the utility gap. Rather, we find that the higher accuracy of ScatterNet classifiers is due to their faster convergence rate *when trained without noise*.

Table 6.8: **Number of trainable parameters of our models.** For CIFAR-10, we consider two different end-to-end CNN architectures (see Table 6.4), the smaller of which has approximately as many parameters as the linear ScatterNet model.

	MNIST & Fashion-MNIST	CIFAR-10
ScatterNet+Linear	40K	155K
ScatterNet+CNN	33K	187K
CNN	26K	551K / 168K

Table 6.9: **Comparison of small and large CIFAR-10 CNNs.** Test accuracy (in %) for two different model sizes on CIFAR-10 for a DP budget of $(\epsilon = 3, \delta = 10^{-5})$. We compare two variants of the end-to-end CNN architecture from Table 6.4, with respectively 551K and 168K parameters. Average and standard deviation computed over five runs.

Model	Parameters	Accuracy
CNN	168K	60.7 ± 0.3
	551K	59.2 ± 0.1

6.3.1 Smaller Models Are Not Easier to Train Privately

The utility of private learning typically degrades as the model’s dimensionality increases [11, 34]. This is also the case with DP-SGD which adds Gaussian noise, of scale proportional to the gradients, to *each* model parameter. We thus expect smaller models to be easier to train privately. Yet, as we see from Table 6.8, for MNIST and Fashion-MNIST the linear ScatterNet model has *more* parameters than the CNNs. For CIFAR-10, the end-to-end CNN we used is larger, so we repeat the experiment from Section 6.2 with a CNN of comparable size to the ScatterNet classifiers.

Specifically, we take the end-to-end CIFAR-10 CNN architecture from Table 6.4 and reduce the number of filters in each convolutional layer by a factor of two and remove the last convolutional layer). This results in a CNN model with a comparable number of trainable parameters as the linear ScatterNet classifier (see Table 6.8). In Table 6.9, we compare the privacy-utility of this smaller CNN models with the original larger CNN model evaluated in Section 6.2. While the change of model architecture does affect the model accuracy, the effect is minor, and the accuracy remains far below that of the ScatterNet classifiers with a comparable number of parameters.

Thus, the dimensionality of ScatterNet classifiers fails to explain their better performance.

6.3.2 Models With Handcrafted Features Converge Faster *Without Privacy*

DP-SGD typically requires a smaller learning rate than noiseless (clipped) SGD, so that the added noise gets averaged out over small steps. We indeed find that the optimal learning rate when training with DP-SGD is an order of magnitude lower than the optimal learning rate for training without

noise addition (with gradients clipped to the same norm in both cases).

To understand the impact of gradient noise on the learning process, we conduct the following experiment: we select a low learning rate that is near-optimal for training models with gradient noise, and a high learning rate that is near-optimal for training without noise. For both learning rates, we train models both with and without noise. When training without privacy, we still clip gradients to a maximal norm of $C = 0.1$, but omit the noise addition step of DP-SGD (and we also omit the noise when using Data Normalization). The hyper-parameters for this experiment are in the table below.

Dataset	Batch size B	Gradient Norm C	Learning rate η (low, high)	Epochs T	Normalization
MNIST	512	0.1	($1/2$, 8)	40	Data Norm. ($\sigma_{\text{norm}} = 8$)
Fashion-MNIST	512	0.1	(1, 16)	40	Group Norm. ($G = 81$)
CIFAR-10	512	0.1	($1/4$, 4)	60	Data Norm. ($\sigma_{\text{norm}} = 8$)

Figure 6.3 shows that with a high learning rate, all classifiers converge rapidly when trained without noise, but gradient noise vastly degrades performance. With a low learning rate however, training converges similarly whether we add noise or not. What distinguishes the ScatterNet models is the faster convergence rate of *noiseless* SGD. Thus, we find that handcrafted features are beneficial for private learning because they result in a simpler learning task where training converges rapidly even with small update steps.

Our analysis suggests two avenues towards obtaining higher accuracy with private deep learning:

- *Faster convergence*: Figure 6.3 suggests that faster convergence of *non-private* training could translate to better private learning. DP-SGD with adaptive updates (e.g., Adam [134]) indeed sometimes leads to small improvements [35, 198, 288]. Investigating private variants of *second-order optimization methods* is an interesting direction for future work.
- *More training steps (a.k.a more data)*: For a fixed DP-budget ϵ and noise scale σ , increasing the training set size N allows for running more steps of DP-SGD [166]. In Section 6.4.1, we investigate how the collection of additional private data impacts the utility of private end-to-end models.

6.4 Towards Better Private Deep Learning

We have shown that on standard vision tasks, private learning strongly benefits from *handcrafted features*. Further improving our private baselines seems hard, as they come close to the maximal accuracy of ScatterNet models (see Table 6.2). We thus turn to other avenues for obtaining stronger privacy-utility guarantees. We focus on CIFAR-10, and discuss two natural paths towards better

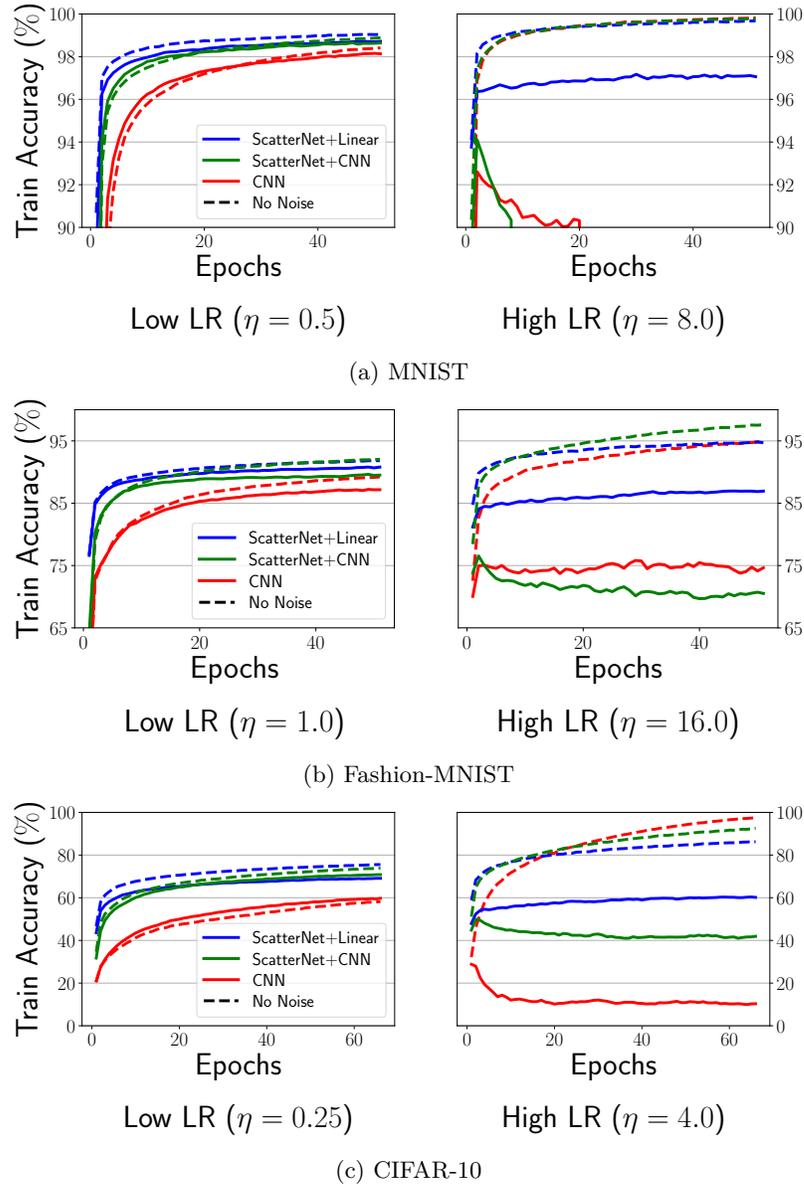


Figure 6.3: **Convergence rates of private and non-private models.** Compares the convergence rates of linear classifiers fine-tuned on ScatterNet features, CNNs fine-tuned on ScatterNet features, and end-to-end CNNs with and without noise addition in DP-SGD. (Left): low learning rate. (Right): high learning rate.

private models: (1) access to a larger *private* training set, and (2) access to a *public* image dataset from a different distribution (some works also consider access to public unlabeled data from the same distribution as the private data [193, 196, 291]).

6.4.1 Improving Privacy by Collecting More Data

We first analyze the benefits of additional *private labeled data* on the utility of private models. Since the privacy budget consumed by DP-SGD scales inversely with the size of the training data N , collecting more data allows either to train for more steps, or to lower the amount of noise added per step—for a fixed DP budget ϵ .

Experimental setup. To obtain a larger dataset comparable to CIFAR-10, we use an additional 500K images from the Tiny Images dataset [249],⁷ which were collected and labeled by Carmon et al. [33] using a pre-trained CIFAR-10 classifier⁸ (see [33, Appendix B.6] for details on the selection process for this dataset). We create datasets of size $N \in \{10K, 25K, 50K, 100K, 250K, 550K\}$ by taking subsets of this larger dataset. We only use the data of Carmon et al. [33] to complement the CIFAR-10 dataset when $N > 50K$. As noted by Carmon et al. [33], the additional 500K images do not entirely match the distribution of CIFAR-10. Nevertheless, we find that training our classifiers *without privacy* on augmented datasets of size $N > 50K$ does not negatively impact the test accuracy on CIFAR-10.

For each training set size, we re-train our models with a hyper-parameter search. To limit computational cost, and informed by our prior experiments, we fix all parameters except for the learning rate and number of epochs (normalized by the size of the original CIFAR-10 data). When applying Data Normalization to ScatterNet features, we compute the per-channel statistics only over the original CIFAR-10 samples, and compute the privacy guarantees of `PrivDataNorm` using the Rényi DP analysis of the sampled Gaussian mechanism [171, 269]. The list of hyper-parameters is in the table below.

⁷The full Tiny Images dataset was recently withdrawn by its curators, following the discovery of a large number of offensive class labels [204]. The subset collected by Carmon et al. [33] contains images that most closely match the original CIFAR-10 labels, and is thus unlikely to contain offensive content.

⁸The privacy guarantees obtained with this dataset could be slightly overestimated, as the pseudo-labels of Carmon et al. [33] are obtained using a model pre-trained on CIFAR-10, thus introducing dependencies between private data points.

Parameter	Value for dataset of size N
DP guarantee (ϵ, δ)	$(3, 1/2N)$
Gradient norm C	0.1
Momentum	0.9
Batch size B	8192
Learning rate η	$\{1/8, 1/4, 1/2, 1, 2\} \cdot 8192/512$
Epochs T	$\{15, 30, 60, 120\} \cdot 50000/N$
Data Norm. params $(C_1, C_2, \sigma_{\text{norm}})$	$(1, 1.5, 8)$

Results. The optimal values we found for these parameters are given in the table below.

N	ScatterNet+Linear		ScatterNet+CNN		CNN	
	Epochs T	Learning rate η	Epochs T	Learning rate η	Epochs T	Learning rate η
10K	30	$1/8$	60	$1/8$	30	$1/8$
25K	30	$1/4$	60	$1/8$	60	$1/8$
50K	60	$1/4$	60	$1/4$	60	$1/4$
100K	60	$1/2$	120	$1/4$	120	$1/4$
250K	120	$1/2$	120	1	120	1
550K	120	1	120	1	120	1

As we increase the dataset size, we obtain better accuracy by training for more steps and with higher learning rates. Figure 6.4 reports the final accuracy for these best-performing models.

We find that we need about *an order-of-magnitude increase in the size of the private training dataset* in order for end-to-end CNNs to outperform ScatterNet features. As shown above, larger datasets allow DP-SGD to be run for more steps at a fixed privacy budget and noise level (as also observed in [166])—thereby overcoming the slow convergence rate we uncovered in Section 6.3. While the increased sample complexity of private deep learning might be viable for “internet-scale” applications (e.g., language modeling across mobile devices), it is detrimental for sensitive applications with more stringent data collection requirements, such as in healthcare.

6.4.2 Transfer Learning: Better Features from Public Data

Transfer learning is a natural candidate for privacy-preserving computer vision, as features learned on public image data often significantly outperform handcrafted features [210]. We consider two transfer learning settings. The first transfers from CIFAR-100 to CIFAR-10, where the labeled CIFAR-100 data is assumed public. The second transfer from public *unlabeled* ImageNet to CIFAR-10.

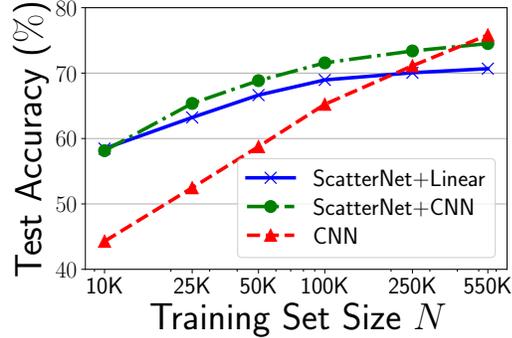


Figure 6.4: **Trade-off between model accuracy and the size of the training set.** Shows the CIFAR-10 test accuracy for a training set of size N and a DP budget of $(\epsilon = 3, \delta = 1/2N)$. For $N > 50K$, we augment CIFAR-10 with pseudo-labeled Tiny Images collected by Carmon et al. [33].

Experimental Setup. We use a ResNeXt-29 [275] model pre-trained on CIFAR-100,⁹ and a ResNet-50 model trained on unlabeled ImageNet [61] using SimCLRv2 [43].¹⁰

To train private linear classifiers on CIFAR-10, we first extract features from the penultimate layer of the above pre-trained models. For the ResNeXt model, we obtain features of dimension 1024, and for the SimCLRv2 ResNet, we obtain features of dimension 4096. We then use DP-SGD with a similar setup as for the linear ScatterNet classifiers, except that we do not normalize the extracted features. We also target a tighter privacy budget of $(\epsilon = 2, \delta = 10^{-5})$. We then run a hyper-parameter search as listed in the table below. We further report the set of hyper-parameters that resulted in the maximal accuracy for the targeted privacy budget of $(\epsilon = 2, \delta = 10^{-5})$.

Parameter	Values	Best for ResNeXt	Best for SimCLRv2
DP guarantee (ϵ, δ)	$(2, 10^{-5})$	-	-
Gradient norm C	0.1	-	-
Momentum	0.9	-	-
Batch size B	$\{512, 1024, \dots, 16384\}$	2048	1024
Learning rate η	$\{1/2, 1, 2, 4\} \cdot B/512$	$2 \cdot 2048/512$	$2 \cdot 1024/512$
Epochs T	$\{15, 25, 40\}$	40	40

Results. Figure 6.5 shows the best test accuracy achieved for each DP budget, averaged across five runs.

With features transferred from the CIFAR-100 ResNeXt model, a non-private linear model trained on transferred features achieves 84% accuracy on CIFAR-10. With DP-SGD, we reach an accuracy of 80.0% at a budget of $(\epsilon = 2, \delta = 10^{-5})$, a significant improvement over prior work for

⁹<https://github.com/bearpaw/pytorch-classification>. Accessed 2021-6-22.

¹⁰<https://github.com/google-research/simclr>. Accessed 2021-6-22.

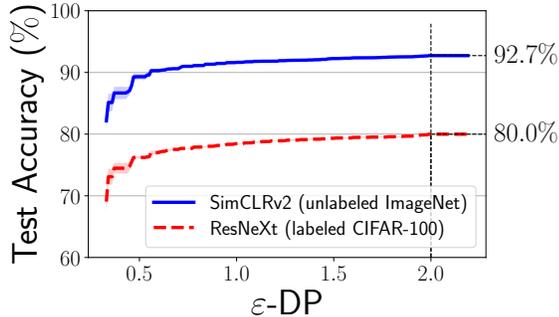


Figure 6.5: **Privacy-utility tradeoffs for transfer learning on CIFAR-10.** We fine-tune linear models on features from a ResNeXt model trained on CIFAR-100, and from a SimCLR model trained on unlabeled ImageNet.

the same setting and privacy budget, e.g., 67% accuracy in [1] and 72% accuracy in [197]. The large gap between our results and prior work is mainly attributed to a better choice of *source model* (e.g., the transfer learning setup in [197] achieves 75% accuracy on CIFAR-10 *in the non-private setting*). Mirroring the work of Kornblith et al. [139] on non-private transfer learning, we thus find that the heuristic rule “better models transfer better” also holds with differential privacy.

With the features transferred from unlabeled ImageNet with SimCLRv2, a non-private linear model achieves 95% accuracy on CIFAR-10 (using *labeled* ImageNet data marginally improves non-private transfer learning to CIFAR-10 [42]). With DP-SGD, we train a linear model to 92.7% accuracy for a DP budget of $(\epsilon = 2, \delta = 10^{-5})$.

6.5 Additional Experiments

6.5.1 On the Effect of Batch Sizes in DP-SGD

In this section, we revisit the question of the selection of an optimal batch size for DP-SGD. In their seminal work, Abadi et al. [1] already investigated this question, and noted that the choice of batch size can have a large influence on the privacy-utility tradeoff. They empirically found that for a dataset of size N , a batch size of size approximately \sqrt{N} produced the best results. However, their experiments measured the effect of the batch size while keeping other parameters, including the noise multiplier σ and the learning rate η , *fixed*.

When training without privacy, it has been shown empirically that the choice of batch size has little effect on the convergence rate of SGD, *as long as the learning rate η is scaled linearly with the batch size* [97]. Hereafter, we argue formally and demonstrate empirically that if we use a linear learning rate scaling, and fix the number of training epochs T for a target privacy budget ϵ , then the choice of batch size also has a minimal influence on the performance of DP-SGD.

We first consider the effect of the sampling rate B/N on the noise scale σ required to attain a fixed privacy budget of ε after T epochs. There is no known closed form expression for σ , so it is usually estimated numerically. We empirically establish the following claim, and verify numerically that it holds for our setting in Figure 6.6:

Claim 6.6. *Given a fixed DP budget (ε, δ) to be reached after T epochs, the noise scale σ as a function of the sampling rate B/N is given by $\sigma(B/N) \approx c \cdot \sqrt{B/N}$, for some constant $c \geq 0$.*

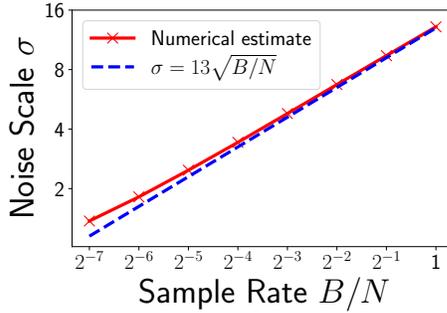


Figure 6.6: **Noise scale as a function of the sample rate.** Plots the noise scale σ for DP-SGD that results in a privacy guarantee of $(\varepsilon = 3, \delta = 10^{-5})$ after 60 training epochs, for different batch sampling rates B/N .

Given this relation between batch size and noise scale, we proceed with a similar analysis as in [97], for the case of DP-SGD. Given some initial weight θ_t , performing k steps of DP-SGD with clipping norm $C = 1$, batch size B , learning rate η and noise scale σ yields:

$$\begin{aligned} \theta_{t+k} &= \theta_t - \eta \sum_{j < k} \frac{1}{B} \left(\sum_{x \in \mathbf{B}_{t+j}} \tilde{g}_{t+j}(x) + \mathcal{N}(0, \sigma^2 \mathbf{I}) \right) \\ &= \left(\theta_t - \eta \frac{1}{B} \sum_{j < k} \sum_{x \in \mathbf{B}_{t+j}} \tilde{g}_{t+j}(x) \right) + \mathcal{N}\left(0, \frac{k\eta^2 \sigma^2}{B^2} \mathbf{I}\right) \end{aligned}$$

If we instead take a single step of DP-SGD with larger batch size kB , a linearly scaled learning rate of $k\eta$, and an adjusted noise scale $\tilde{\sigma} = \sqrt{k}\sigma$ (by Claim 6.6), we get:¹¹

$$\begin{aligned} \theta_{t+1} &= \theta_t - k\eta \frac{1}{kB} \left(\sum_{j < k} \sum_{x \in \mathbf{B}_{t+j}} \tilde{g}_t(x) + \mathcal{N}(0, \tilde{\sigma}^2 \mathbf{I}) \right) \\ &= \left(\theta_t - \eta \frac{1}{B} \sum_{j < k} \sum_{x \in \mathbf{B}_{t+j}} \tilde{g}_t(x) \right) + \mathcal{N}\left(0, \frac{k\eta^2 \sigma^2}{B^2} \mathbf{I}\right) \end{aligned}$$

Thus, we find that the total noise in both updates is identical. Under the same heuristic assumption

¹¹We make a small simplification to our analysis here and assume that one batch of DP-SGD sampled with selection probability $\frac{kB}{N}$ is identical to k batches sampled with selection probability $\frac{B}{N}$.

as in [97] that $\tilde{g}_t(x) \approx \tilde{g}_{t+j}(x)$ for all $j < k$, the two DP-SGD updates above are thus similar. This analysis suggests that as in the non-private case [97], increasing the batch size and linearly scaling the learning rate should have only a small effect on a model’s learning curve.

We now verify this claim empirically. We follow the experimental setup in Section 6.2, and set a privacy budget of $(\epsilon = 3, \delta = 10^{-5})$ to be reached after a fixed number of epochs T . For different choices of batch size B , we numerically compute the noise scale σ that fits this “privacy schedule”. For the initial batch size of $B_0 = 512$, we select a base learning rate η that maximizes test accuracy at epoch T . As we increase the batch size to $B = kB_0$, we linearly scale the learning rate to $k\eta$. The concrete parameters are given below:

	Epochs T	Batch size B	Learning rate η
MNIST	40	{512, 1024, 2048, 4096}	$1/2 \cdot B/512$
Fashion-MNIST	40	{512, 1024, 2048, 4096}	$1 \cdot B/512$
CIFAR-10	60	{512, 1024, 2048, 4096}	$1/4 \cdot B/512$

As we can see in Figure 6.7, the training curves for CNNs trained with DP-SGD are indeed near identical across a variety of batch sizes.

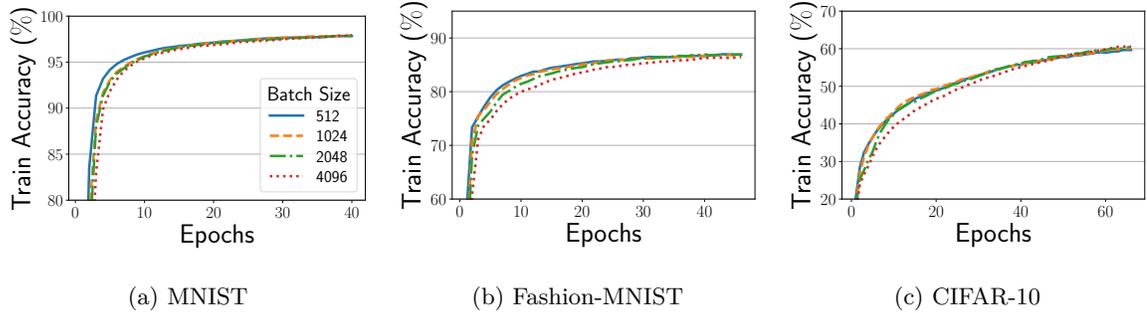


Figure 6.7: **Convergence rate of DP-SGD for different batch sizes.** Plots the training accuracy for a fixed targeted privacy budget of $(\epsilon = 3, \delta = 10^{-5})$ after $T = 40$ or $T = 60$ epochs, and linear scaling of the learning rate $\eta \cdot B/512$.

6.5.2 Comparing DP-SGD and Privacy Amplification by Iteration

While DP-SGD is the algorithm of choice for differentially private *non-convex* learning, it is unclear why it should be the best choice for learning private *linear models*. Indeed, starting with the work of Chaudhuri et al. [34], there have been many other proposals of algorithms for private convex optimization with provable utility guarantees, e.g., [11, 78, 133]. Yet, Yu et al. [282] show that DP-SGD can achieve higher utility than many of these approaches, both asymptotically and empirically.

Here, we take a closer look at the “Privacy Amplification by Iteration” work of [78]. Feldman et al. [78] observe that DP-SGD guarantees differential privacy for *every* gradient update step. Under

the assumption that intermediate model updates can be hidden from the adversary, they propose a different analysis of DP-SGD for convex optimization problems that has a number of conceptual advantages. First, the algorithm of Feldman et al. [78] does not require the training indices selected for each batch \mathbf{B}_t do be hidden from the adversary. Second, their approach can support much smaller privacy budgets than DP-SGD.

However, we show that these benefits come at a cost in practice: for the range of privacy budgets we consider in this work, DP-SGD requires adding *less* noise than Privacy Amplification by Iteration (PAI). To compare the two approaches, we proceed as follows: We analytically compute the noise scale σ that results in a privacy guarantee of $(\epsilon, \delta = 10^{-5})$ after 10 training epochs with a batch sampling rate of $512/50000$.¹² Figure 6.8 shows that DP-SGD requires adding less noise, except for large privacy budgets ($\epsilon > 40$), or very small ones ($\epsilon < 0.2$). In the latter case, both algorithms require adding excessively large amounts of noise. We observe a qualitatively similar behavior for other sampling rates.

For completeness, we evaluate the PAI algorithm of Feldman et al. [78] for training linear ScatterNet classifiers on CIFAR-10. We evaluate a broader range of hyper-parameters, including different clipping thresholds $C \in \{0.1, 1, 10\}$ (PAI clips the data rather than the gradients), a wider range of batch sizes $B \in \{32, 64, \dots, 2048\}$, and a wider range of base learning rates $\eta \in \{2^{-3}, 2^{-2}, \dots, 2^3\}$. We find that for privacy budgets $1 \leq \epsilon \leq 3$, the optimal hyper-parameters for PAI and DP-SGD are similar, but the analysis of PAI requires a larger noise scale σ . As a result, PAI performs worse than DP-SGD, as shown in Figure 6.9.

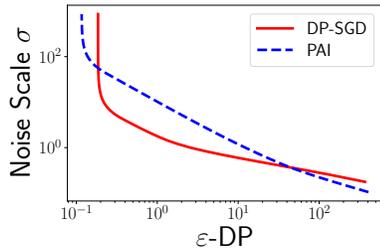


Figure 6.8: **Comparison of noise scales for DP-SGD and Privacy Amplification by Iteration.** Plots the noise scale σ required for a privacy guarantee of $(\epsilon, \delta = 10^{-5})$ after 10 training epochs with batch sampling rate $512/50000$. Privacy Amplification by Iteration (PAI) [78] requires less noise than DP-SGD only for very small or very large privacy budgets.

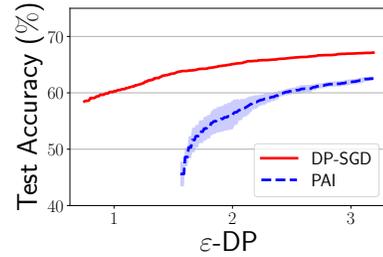


Figure 6.9: **Privacy-accuracy tradeoffs for DP-SGD [1] and Privacy Amplification by Iteration (PAI) [78].** Shows the maximum accuracy achieved for each privacy budget when training a private linear ScatterNet classifier on CIFAR-10, averaged over five runs.

¹²The guarantees of Privacy Amplification by Iteration apply unevenly to the elements of the training data. We choose the noise scale so that at least 99% of the data elements enjoy (ϵ, δ) -DP.

6.5.3 DP-SGD With Poisson Sampling

The analysis of DP-SGD [1, 171] assumes that each batch \mathbf{B}_t is created by independently selecting each training sample with probability B/N . This is in contrast to typical implementations of SGD, where the training data is randomly shuffled once per epoch, and divided into successive batches of size *exactly* B . The latter “random shuffle” approach has been used in most implementations of DP-SGD (e.g., in `TensorFlow/privacy`¹³ and `Opacus`¹⁴) as well as in prior work (e.g., [1, 198]), with the (implicit) assumption that this difference in batch sampling strategies will not affect model performance. We verify that this assumption is indeed valid in our setting. We re-train the linear ScatterNet and end-to-end CNN models that achieved the highest accuracy for a DP budget of ($\epsilon = 3, \delta = 10^{-5}$) (with the best hyper-parameters detailed in Section 6.2.4), using the correct “Poisson sampling” strategy. The test accuracy of these models (averaged over five runs) are shown in Table 6.10. For all datasets and models, the two sampling schemes achieve similar accuracy when averaged over five runs.

Table 6.10: **Comparison of DP-SGD with Poisson sampling and random shuffling.** Shows model accuracy for two different batch sampling schemes: (1) *Poisson sampling*, where a batch is formed by selecting each data point independently with probability B/N ; (2) *Random shuffle*, where the training set is randomly shuffled at the beginning of each epoch, and split into consecutive batches of size B . For both sampling schemes, we report the best test accuracy (in %) at a DP budget of ($\epsilon = 3, \delta = 10^{-5}$), with means and standard deviations over five runs.

Dataset	ScatterNet		CNN	
	Poisson Sampling	Random Shuffle	Poisson Sampling	Random Shuffle
MNIST	98.6 \pm 0.1	98.7 \pm 0.0	98.0 \pm 0.1	98.1 \pm 0.0
Fashion-MNIST	89.6 \pm 0.1	89.7 \pm 0.0	86.1 \pm 0.2	86.0 \pm 0.1
CIFAR-10	66.8 \pm 0.2	67.0 \pm 0.0	59.0 \pm 0.4	59.2 \pm 0.1

6.6 Conclusion and Open Problems

We have demonstrated that differentially private learning benefits from “handcrafted” features that encode priors on the learning task’s domain. In particular, we have shown that private ScatterNet classifiers outperform end-to-end CNNs on MNIST, Fashion-MNIST and CIFAR-10. We have further found that handcrafted features can be surpassed when given access to more data, either a larger private training set, or a public dataset from a related domain. In addition to introducing strong baselines for evaluating future improvements to private deep learning and DP-SGD, our results suggest a number of open problems and directions for future work:

¹³<https://github.com/tensorflow/privacy>. Accessed 2021-6-22.

¹⁴<https://github.com/pytorch/opacus>. Accessed 2021-6-22.

Improving DP by accelerating convergence. Our analysis in Section 6.3 shows that a limiting factor of private deep learning is the slow convergence rate of end-to-end deep models. While the existing literature on second-order optimization for deep learning has mainly focused on improving the overall *wall-clock time* of training, it suffices for DP to reduce the number of private *training steps*—possibly at an increase in computational cost.

Federated learning. While we have focused on a standard centralized setting for DP, our techniques can be extended to decentralized training schemes such as Federated Learning [19, 128, 165]. DP has been considered for Federated Learning [86, 166], but has also been found to significantly degrade performance in some settings [284].

Handcrafted features for ImageNet and non-vision domains. To our knowledge, there have not yet been any attempts to train ImageNet models with DP-SGD, partly due to the cost of computing per-sample gradients. While linear classifiers are unlikely to be competitive on ImageNet, handcrafted features can also help private learning by accelerating the convergence of CNNs, as we have shown in Figure 6.1. Notably, Oyallon et al. [188] match the (non-private) accuracy of AlexNet [142] on ImageNet with a small six-layer CNN trained on ScatterNet features. Another interesting direction is to extend our results to domains beyond vision, e.g., with handcrafted features for text [162] or speech [3].

Chapter 7

Slalom: Faster Private Inference With Trusted Hardware

Once a machine learning (ML) model has been trained (possibly with differential privacy as in Chapter 6), it is routinely *outsourced* to a remote server to compute client predictions. Prominent examples include cloud-based ML APIs (e.g., a speech-to-text application that consumes user-provided data) or general ML-as-a-Service platforms. The outsourcing of these ML services raises natural concerns for the *integrity* and *privacy* of the client’s predictions.

Trusted Execution Environments (TEEs), e.g, Intel SGX [164], ARM TrustZone [2] or Sanctum [56] offer a pragmatic solution to this problem. TEEs use hardware and software protections to isolate sensitive code from other applications, while *attesting* to its correct execution. Running outsourced ML computations in TEEs provides remote clients with strong privacy and integrity guarantees.

For outsourced ML computations, TEEs outperform pure cryptographic approaches (e.g. [87, 88, 126, 174]) by multiple orders of magnitude. At the same time, the isolation guarantees of TEEs still come at a steep price in performance, compared to untrusted alternatives (i.e., running ML models on contemporary hardware with no security guarantees). For instance, Intel SGX [116] incurs significant overhead for memory intensive tasks [104, 186], has difficulties exploiting multi-threading, and is currently limited to desktop CPUs that are outmatched by untrusted alternatives (e.g., GPUs or server CPUs). Thus, our thesis is that *for modern ML workloads, TEEs will be at least an order of magnitude less efficient than the best available untrusted hardware.*

This leads us to the main question of this chapter:

How can we most efficiently leverage TEEs for secure machine learning?

This was posed by Stoica et al. [242] as one of nine open research problems for system challenges in AI. A specific challenge they raised is that of appropriately splitting ML computations between

trusted and untrusted components, to increase efficiency as well as security by minimizing the Trusted Computing Base.

In this chapter, we explore a novel approach to this challenge, wherein a neural network’s execution is *partially outsourced from a TEE to a co-located, untrusted but faster device*. Our approach, inspired by the verifiable ASICs of Wahby et al. [267], differs from cryptographic ML outsourcing. In our case, work is delegated between two *co-located* parties, thus allowing for highly interactive—yet conceptually simpler—outsourcing protocols with orders-of-magnitude better efficiency. Our approach also departs from prior systems that execute neural networks fully in a TEE [45, 103, 111, 183].

The main observation that guides our approach is that matrix multiplication—the main bottleneck in neural networks—admits a concretely efficient verifiable outsourcing scheme known as Freivalds’ algorithm [82], which can also be turned private in our setting. Our TEE selectively outsources these CPU intensive steps to a fast untrusted co-processor (and runs the remaining steps itself) therefore achieving much better performance than running the entire computation in the enclave, without compromising security.

We propose Slalom, a framework for efficient neural network inference in *any* trusted execution environment (e.g., SGX or Sanctum). To evaluate Slalom, we build a lightweight neural network library for Intel SGX, which may be of independent interest. Our library allows for outsourcing all linear layers to an untrusted GPU without compromising integrity or privacy.

We formally prove Slalom’s security, and evaluate it on multiple canonical ML models with a variety of computational costs—VGG16 [236], MobileNet [109], and ResNets [105]. Compared to running all computations in SGX, outsourcing linear layers to an untrusted GPU increases throughput (as well as energy efficiency) by $6\times$ to $20\times$ for verifiable inference, and by $4\times$ to $11\times$ for verifiable and private inference. Finally, we discuss open challenges towards efficient verifiable training of neural networks in TEEs.

7.1 Background

7.1.1 Problem Setting

We consider an outsourcing scheme between a client \mathcal{C} and a server \mathcal{S} , where \mathcal{S} executes a classifier $f(x) : \mathbb{R}^d \rightarrow [C]$ on data provided by \mathcal{C} . The classifier can either belong to the user (e.g., as in some ML-as-a-service platforms), or to the server (e.g., as in a cloud-based ML API). Depending on the application, this scheme should satisfy one or more of the following security properties (see Section 7.2 for formal definitions):

- **t-Integrity:** For any \mathcal{S} and input x , the probability that a user interacting with \mathcal{S} does not abort (i.e., output \perp) and outputs an incorrect value $\tilde{y} \neq f(x)$ is less than t .
- **Privacy:** The server \mathcal{S} learns no information about the user’s input x .

- **Model privacy:** If the model f is provided by the user, \mathcal{S} learns no information about f (beyond e.g., its approximate size). If f belongs to the server, \mathcal{C} learns no more about f than what is revealed by $y = f(x)$.¹

7.1.2 Trusted Execution Environments (TEEs), Intel SGX, and a Strong Baseline

Trusted Execution Environments (TEE) such as Intel SGX, ARM TrustZone or Sanctum [56] enable execution of programs in *secure enclaves*. Hardware protections isolate computations in enclaves from all programs on the same host, including the operating system. Enclaves can produce *remote attestations*—digital signatures over an enclave’s code—that a remote party can verify using the manufacturer’s public key. Our experiments with Slalom use hardware enclaves provided by Intel SGX.

Details on Intel SGX. SGX enclaves isolate execution of a program from all other processes on a same host, including a potentially malicious OS. In particular, enclave memory is fully encrypted and authenticated. When a word is read from memory into a CPU register, a Memory Management Engine handles the decryption [55].

While SGX covers many software and hardware attack vectors, there is a large and prominent class of side-channel attacks that it explicitly does not address [55, 254]. In the past years, many attacks have been proposed, with the goal of undermining privacy of enclave computations [20, 96, 147, 173, 263, 278]. Most of these attacks rely on data dependent code behavior in an enclave (e.g., branching or memory access) that can be partially observed by other processes running on the same host. These side-channels are a minor concern for the neural network computations considered in this paper, as the standard computations in a neural network are data-oblivious (i.e., the same operations are applied regardless of the input data) [183].

The recent Spectre attacks on speculative execution [135] also prove damaging to SGX (as well as to most other processors), as recently shown [36, 59, 264]. Mitigations for these side-channel attacks are being developed [40, 117, 232, 233] but a truly secure solution might require some architectural changes, e.g., as in the proposed Sanctum processor [56].

We refrain from formally modeling SGX’s (or other TEE’s) security in this paper, as Slalom is mostly concerned with outsourcing protocols wherein the TEE acts as a client. We refer the interested reader to [80, 200, 245] for different attempts at such formalisms.

A baseline for outsourcing ML with TEEs. TEEs offer an efficient solution to the ML outsourcing problem:

¹For this zero-knowledge guarantee to be meaningful in our context, \mathcal{S} would first *commit* to a specific model, and then convince \mathcal{C} that this model was correctly evaluated on her input, without revealing anything else about the model.

The server runs an enclave that initiates a secure communication with C and evaluates a model f on C 's input data.

This simple scheme (which we implemented in SGX, see Section 7.4) outperforms cryptographic ML outsourcing protocols by 2-3 orders of magnitude (albeit under a different trust model). See Table 7.1 for a comparison to two representative works, SafetyNets [87] and Gazelle [126].

Yet, SGX's security comes at a performance cost, and there remains a large gap between TEEs and untrusted devices. For example, current SGX CPUs are limited to 128 MB of *Processor Reserved Memory* (PRM) [55] and incur severe paging overheads when exceeding this allowance [186]. We also failed to achieve noticeable speed ups for multi-threaded neural network evaluations in SGX enclaves (see Section 7.4.6). For neural network computations, current SGX enclaves thus cannot compete—in terms of performance or energy efficiency (see section 7.4.4, A note on energy efficiency)—with contemporary *untrusted* hardware, such as a GPU or server CPU.

In this work, we treat the above simple (yet powerful) TEE scheme as a baseline, and identify settings where we can still improve upon it. We will show that our system, Slalom, substantially outperforms this baseline when the server has access to the model f (e.g., f belongs to \mathcal{S} as in cloud ML APIs, or f is public). Slalom performs best for verifiable inference (the setting considered in SafetyNets [87]). If the TEE can run some offline data-independent preprocessing (e.g., as in Gazelle [126]), Slalom also outperforms the baseline for *private* (and verifiable) outsourced computations in a later online phase. Such a two-stage approach is viable if user data is sent at irregular intervals yet has to be processed with high throughput when available.

Performance comparison of neural network outsourcing schemes. We provide a brief overview of the outsourcing approaches compared in Table 7.1. Our baseline runs a neural network in a TEE (a single-threaded Intel SGX enclave) and can provide all the security guarantees of an ML outsourcing scheme. On a high-end GPU (an Nvidia TITAN XP), we achieve over $50\times$ higher throughput but no security. For example, for MobileNet, the enclave evaluates 16 images/sec and the GPU 900 images/sec ($56\times$ higher).

SafetyNets [87] and Gazelle [126] are two representative works that achieve respectively integrity and privacy using purely cryptographic approaches (without a TEE). SafetyNets does not hide the model from either party, while Gazelle leaks some architectural details to the client. The cryptographic techniques used by these systems incur large computation and communication overheads in practice. The largest model evaluated by SafetyNets is a 4-layer TIMIT model with quadratic activations which runs at about 13 images/sec (on a notebook CPU). In our baseline enclave, the same model runs at over 3,500 images/sec. The largest model evaluated by Gazelle is an 8-layer CIFAR-10 model. In the enclave, we can evaluate 450 images/sec whereas Gazelle evaluates a single image in 3.5 sec with 300MB of communication between client and server.

Table 7.1: Security guarantees and performance (relative to baseline) of different ML outsourcing schemes.

Approach	TEE	Integrity	Privacy	Model Privacy		Throughput (relative)
				w.r.t. Server	w.r.t. Client	
SafetyNets [87]	-	●	○	○	○	$\leq 1/200 \times$
Gazelle [126]	-	○	●*	○	◐	$\leq 1/1000 \times$
Secure baseline (run model in TEE)	✓	●	●	●	●	$1 \times$
Insecure baseline (run model on GPU)	-	○	○	○	◐	$\geq 50 \times$
Slalom (Ours)	✓	●	●*	○	●	$4 \times - 20 \times$

* With an offline preprocessing phase.

7.1.3 Outsourcing Outsourced Neural Networks and Freivalds’ Algorithm

Our idea for speeding up neural network inference in TEEs is to *further outsource* work from the TEE to a *co-located faster untrusted processor*. Improving upon the above baseline thus requires that the *combined* cost of doing work on the untrusted device and verifying it in the TEE be cheaper than evaluating the full model in the TEE.

Wahby et al. [267, 268] aim at this goal for *arbitrary computations* outsourced between co-located ASICs. The generic non-interactive proofs they use for integrity are similar to those used in SafetyNets [87], which incur overheads that are too large to warrant outsourcing in our setting (e.g., Wahby et al. [267] find that the technology gap between trusted and untrusted devices needs to be of over two decades for their scheme to break even). Similarly for privacy, standard cryptographic outsourcing protocols (e.g., [126]) are unusable in our setting as simply running the computation in the TEE is much more efficient (see Table 7.1).

To overcome this barrier, we design outsourcing protocols tailored to neural networks, leveraging two insights:

1. In our setting, the TEE is *co-located* with the server’s faster untrusted processors, thus widening the design space to *interactive* outsourcing protocols with high communication but better efficiency.
2. The TEE always has knowledge of the model and can *selectively* outsource part of the neural network evaluation and compute others—for which outsourcing is harder—itsself.

Neural networks are a class of functions that are particularly well suited for selective outsourcing. Indeed, non-linearities—which are hard to securely outsource (with integrity or privacy)—represent a small fraction of the computation in a neural network so we can evaluate these in the TEE (e.g., for VGG16 inference on a single CPU thread, about 1.5% of the computation is spent on non-linearities).

In contrast, linear operators—the main computational bottleneck in neural networks—admit for a conceptually simple yet concretely efficient secure delegation scheme, described below.

Integrity. We verify integrity of outsourced linear layers using variants of an algorithm by Freivalds [82].

Lemma 7.1 (Freivalds). *Let A, B and C be $n \times n$ matrices over a field \mathbb{F} and let s be a uniformly random vector in \mathbb{S}^n , for $\mathbb{S} \subseteq \mathbb{F}$. Then, $\Pr[Cs = A(Bs) \mid C \neq AB] = \Pr[(C - AB)s = \mathbf{0} \mid (C - AB) \neq \mathbf{0}] \leq 1/|\mathbb{S}|$.*

The randomized check requires $3n^2$ multiplications, a significant reduction (both in concrete terms and asymptotically) over evaluating the product directly. The algorithm has no false negatives and trivially extends to rectangular matrices. Independently repeating the check k times yields soundness error $1/|\mathbb{S}|^k$.

Privacy. Input privacy for outsourced linear operators could be achieved with linearly homomorphic encryption, but the overhead (see the micro-benchmarks in [126]) is too high to compete with our baseline (i.e., computing the function directly in the TEE would be faster than outsourcing it over encrypted data).

We instead propose a very efficient two-stage approach based on symmetric cryptography, i.e., an additive stream cipher. Let $g : \mathbb{F}^m \rightarrow \mathbb{F}^n$ be a linear function over a field \mathbb{F} . In an offline phase, the TEE generates a stream of *one-time-use pseudorandom elements* $r \in \mathbb{F}^m$, and pre-computes $u = g(r)$. Then, in the online phase when the remote client sends an input x , the TEE computes $\text{Enc}(x) = x + r$ over \mathbb{F}^m (i.e., a secure encryption of x with a stream cipher), and outsources the computation of $g(\text{Enc}(x))$ to the faster processor. Given the result $g(\text{Enc}(x)) = g(x + r) = g(x) + g(r) = g(x) + u$, the TEE recovers $g(x)$ using the pre-computed u .

Communication. Using Freivalds’ algorithm and symmetric encryption for each linear layer in a neural network incurs high interaction and communication between the TEE and untrusted co-processor (e.g., over 50MB per inference for VGG16, see Table 7.3). This would be prohibitive if they were not co-located. There are protocols with lower communication than repeatedly using Freivalds’ ([79, 87, 248]). Yet, these incur a high overhead on the prover in practice and are thus not suitable in our setting.

7.2 Formal Security Definitions

We define a secure outsourcing scheme, between a client \mathcal{C} and a server \mathcal{S} , for a classifier $f(x) : \mathbb{R}^d \rightarrow [C]$ from some family \mathcal{F} (e.g., all neural networks of a given size). We first assume that the model f is known to both \mathcal{C} and \mathcal{S} :

Definition 7.2 (Secure Outsourcing Schemes). A secure outsourcing scheme consists of an offline preprocessing algorithm Preproc , as well as an interactive online protocol $\text{Outsource}\langle\mathcal{C}, \mathcal{S}\rangle$, defined as follows:

- $\text{st} \leftarrow \text{Preproc}(f, 1^\lambda)$: The preprocessing algorithm is run by \mathcal{C} and generates some data-independent state st (e.g., cryptographic keys or precomputed values to accelerate the online outsourcing protocol.)
- $[C] \cup \{\perp\} \leftarrow \text{Outsource}\langle\mathcal{C}(f, x, \text{st}), \mathcal{S}(f)\rangle$: The online outsourcing protocol is initiated by \mathcal{C} with inputs (f, x, st) . At the end of the protocol, \mathcal{C} either outputs a value $y \in [C]$ or aborts (i.e., \mathcal{C} outputs \perp).

The properties that we may require from a secure outsourcing scheme are:

- **Correctness:** For any $f \in \mathcal{F}$ and $x \in \mathbb{R}^d$, running $\text{st} \leftarrow \text{Preproc}(f, 1^\lambda)$ and $y \leftarrow \text{Outsource}\langle\mathcal{C}(f, x, \text{st}), \mathcal{S}(f)\rangle$ yields $y = f(x)$.
- **t-Integrity:** For any $f \in \mathcal{F}$, input $x \in \mathcal{X}$ and probabilistic polynomial-time adversary \mathcal{S}^* , the probability that $\tilde{y} = \text{Outsource}\langle\mathcal{C}(f, x, \text{st}), \mathcal{S}^*(f)\rangle$ and $\tilde{y} \notin \{f(x), \perp\}$ is less than t .
- **Input privacy:** For any $f \in \mathcal{F}$, inputs $x, x' \in \mathbb{R}^d$ and probabilistic poly-time adversary \mathcal{S}^* , the views of \mathcal{S}^* in $\text{Outsource}\langle\mathcal{C}(f, x, \text{st}), \mathcal{S}^*(f)\rangle$ and $\text{Outsource}\langle\mathcal{C}(f, x', \text{st}), \mathcal{S}^*(f)\rangle$ are computationally indistinguishable.
- **Efficiency:** The online computation of \mathcal{C} in Outsource should be less than the cost for \mathcal{C} to evaluate $f \in \mathcal{F}$.

Model privacy. In some applications a secure outsourcing scheme may also require to hide the model f from either \mathcal{S} or \mathcal{C} (in which case that party would obviously not take f as input in the above scheme).

Privacy with respect to an adversarial server \mathcal{S}^* (which Slalom does not provide), is defined as the indistinguishability of \mathcal{S}^* 's views in $\text{Outsource}\langle\mathcal{C}(f, x, \text{st}), \mathcal{S}^*\rangle$ and $\text{Outsource}\langle\mathcal{C}(f', x, \text{st}), \mathcal{S}^*\rangle$ for any $f, f' \in \mathcal{F}$.

As noted in Section 7.1.1, a meaningful model-privacy guarantee with respect to \mathcal{C} requires that \mathcal{S} first commit to a specific classifier f , and then convinces \mathcal{C} that her outputs were produced with the same model as all other clients'. We refer the reader to Canetti et al. [25] for formal definitions for such *commit-and-prove* schemes, and to Tramèr et al. [254] who show how to trivially instantiate them using a TEE.

7.3 Slalom

We introduce Slalom, a three-step approach for outsourcing neural networks from a TEE to an untrusted but faster device: (1) Inputs and weights are quantized and embedded in a field \mathbb{F} ; (2) Linear layers are outsourced and verified using Freivalds’ algorithm; (3) Inputs of linear layers are encrypted with a pre-computed pseudorandom stream to guarantee privacy. Figure 7.1 shows two Slalom variants, one to achieve integrity, and one to also achieve privacy.

We focus on feed-forward networks with fully connected layers, convolutions, separable convolutions, pooling layers and activations. Slalom can be extended to other architectures (e.g., residual networks, see Section 7.4.4).

7.3.1 Quantization

The techniques we use for integrity and privacy (Freivalds’ algorithm and stream ciphers) work over a field \mathbb{F} . We thus *quantize* all inputs and weights of a neural network to integers, and embed these integers in the field \mathbb{Z}_p of integers modulo a prime p (where p is larger than all values computed in a neural network evaluation, so as to avoid wrap-around).

As in [102], we convert floating point numbers x to a *fixed-point representation* as $\tilde{x} = \text{FP}(x; l) := \text{round}(2^l \cdot x)$. For a linear layer with kernel W and bias b , we define integer parameters $\tilde{W} = \text{FP}(W, l)$, $\tilde{b} = \text{FP}(b, 2l)$. After applying the layer to a quantized input \tilde{x} , we scale the output by 2^{-l} and re-round to an integer.

For efficiency reasons, we perform integer arithmetic using floats (so-called fake quantization), and choose $p < 2^{24}$ to avoid loss of precision (we use $p = 2^{24} - 3$). For the models we evaluate, setting $l = 8$ for all weights and inputs ensures that all neural network values are bounded by 2^{24} , with less than a 0.5% drop in accuracy (see Table 7.3). When performing arithmetic modulo p , we use double-precision floats, to reduce the number of modular reductions required.

In more detail, to compute inner products, we first cast elements to doubles (as a single multiplication in \mathbb{Z}_p would exceed the range of integers exactly representable as floats). Single or double precision floats are preferable to integer types on Intel architectures due to the availability of much more efficient SIMD instructions, at a minor reduction in the range of exactly representable integers.

In our evaluation, we target a soundness error of 2^{-40} for each layer. This leads to a tradeoff between the number of repetitions k of Freivalds’ check, and the size of the set \mathbb{S} from which we draw random values. One check with $|\mathbb{S}| = 2^{40}$ is problematic, as multiplying elements in \mathbb{Z}_p and \mathbb{S} can exceed the range of integers exactly representable as doubles (2^{53}). With $k = 2$ repetitions, we can set $\mathbb{S} = [-2^{19}, 2^{19}]$. Multiplications are then bounded by $2^{24+19} = 2^{43}$, and we can accumulate 2^{10} terms in the inner-product before needing a modular reduction. In practice, we find that increasing k further (and thus reducing $|\mathbb{S}|$) is not worthwhile, as the cost of performing more inner products trumps the savings from reducing the number of modulus.

TEE ($f, x^{(1)}$)	Slalom with integrity	$\mathcal{S}(f)$
for $i \in [1, n]$ do assert $\text{Freivalds}(y^{(i)}, x^{(i)}, W^{(i)})$ $x^{(i+1)} \leftarrow \sigma(y^{(i)})$ return $y^{(n)}$	$\xrightarrow{x^{(1)}}$ $\xleftarrow{y^{(1)} \dots y^{(n)}}$	for $i \in [1, n]$ do $y^{(i)} \leftarrow x^{(i)} W^{(i)}$ $x^{(i+1)} \leftarrow \sigma(y^{(i)})$
TEE ($f, x^{(1)}$)	Slalom with integrity & privacy	$\mathcal{S}(f)$
Preproc: for $i \in [1, n]$ do $r^{(i)} \leftarrow_{\$} \mathbb{F}^{m_i}$ $u^{(i)} \leftarrow r^{(i)} W^{(i)}$ for $i \in [1, n]$ do $\tilde{x}^{(i)} \leftarrow x^{(i)} + r^{(i)}$ $y^{(i)} \leftarrow \tilde{y}^{(i)} - u^{(i)}$ assert $\text{Freivalds}(y^{(i)}, x^{(i)}, W^{(i)})$ $x^{(i+1)} \leftarrow \sigma(y^{(i)})$ return $y^{(n)}$	$\xrightarrow{\tilde{x}^{(i)}}$ $\xleftarrow{\tilde{y}^{(i)}}$	$\tilde{y}^{(i)} \leftarrow \tilde{x}^{(i)} W^{(i)}$

Figure 7.1: **The Slalom algorithms for verifiable and private neural network inference.** The TEE outsources computation of n linear layers of a model f to the untrusted host server \mathcal{S} . Each linear layer is defined by a matrix $W^{(i)}$ of size $m_i \times n_i$ and followed by an activation σ . All operations are over a field \mathbb{F} . The $\text{Freivalds}(y^{(i)}, x^{(i)}, w^{(i)})$ subroutine performs k repetitions of Freivalds' check (possibly using precomputed values as in Section 7.3.2). The pseudorandom elements $r^{(i)}$ (we omit the PRNG for simplicity) and precomputed values $u^{(i)}$ are used only once.

7.3.2 Verifying Common Linear Operators

We now describe Slalom’s approach to verifying the integrity of outsourced linear layers. We summarize this section’s results in Table 7.2.

Common linear layers. Below we describe some common linear operators used in deep neural networks. For simplicity, we omit additive bias terms, and assume that convolutional operators preserve the spatial height and width of their inputs. Our techniques easily extend to convolutions with arbitrary strides, paddings, and window sizes.

A *fully-connected layer* f_{FC} has kernel W of size $(h_{\text{in}} \times h_{\text{out}})$. For an input x of dimension h_{in} , we have $f_{\text{FC}}(x) = x^\top W$. The cost of the layer is $h_{\text{in}} \cdot h_{\text{out}}$ multiplications.

A *convolutional layer* has kernel W of size $(k \times k \times c_{\text{in}} \times c_{\text{out}})$. On input x of size $(h \times w \times c_{\text{in}})$, $f_{\text{conv}}(x) = \text{Conv}(x; W)$ produces an output of size $(h \times w \times c_{\text{out}})$. A convolution can be seen as the combination of two linear operators: a “patch-extraction” process that transforms the input x into an intermediate input x' of dimension $(h \cdot w, k^2 \cdot c_{\text{in}})$ by extracting $k \times k$ patches, followed by a matrix multiplication with W . The cost of this layer is thus $k^2 \cdot h \cdot w \cdot c_{\text{in}} \cdot c_{\text{out}}$ multiplications.

A *separable convolution* has two kernels, W of size $(k \times k \times c_{\text{in}})$ and W' of size $(c_{\text{in}} \times c_{\text{out}})$. On input x of size $(h \times w \times c_{\text{in}})$, $f_{\text{sep-conv}}(x)$ produces an output of size $(h \times w \times c_{\text{out}})$, by applying a depthwise convolution $f_{\text{dp-conv}}(x)$ with kernel W followed by a pointwise convolution $f_{\text{pt-conv}}(x)$ with kernel W' .

A *depthwise convolution* consists of c_{in} independent convolutions with filters of size $k \times k \times 1 \times 1$, applied to a single input channel, which requires $k^2 \cdot h \cdot w \cdot c_{\text{in}}$ multiplications.

A *pointwise convolution* is simply a matrix product with an input of size $(h \cdot w) \times c_{\text{in}}$, and thus requires $h \cdot w \cdot c_{\text{in}} \cdot c_{\text{out}}$ multiplications.

Freivalds’ algorithm for batches. The most direct way of applying Freivalds’ algorithm to arbitrary linear layers of a neural network is by exploiting batching. Any linear layer $f(x)$ from inputs of size m to outputs of size n can be represented (with appropriate reshaping) as $f(x) = x^\top W$ for a (often sparse and implicit) $m \times n$ matrix W .

For a batch X of size B , we can outsource $f(X)$ and check that the output Y satisfies $f(s^\top X) = s^\top Y$, for a random vector s (we are implicitly applying Freivalds to the matrix product $XW = Y$). As the batch size B grows, the cost of evaluating f is amortized and the total verification cost is $|X| + |Y| + \text{cost}_f$ multiplications (i.e., we approach one operation per input and output). Yet, as we show in Section 7.4.4, while batched verification is worthwhile for processors with larger memory, it is prohibitive in SGX enclaves due to the limited PRM.

For full convolutions (and pointwise convolutions), a direct application of Freivalds’ check is worthwhile even for single-element batches. For $f(x) = \text{Conv}(x, W)$ and purported output y , we can sample a random vector s of dimension c_{out} (the number of output channels), and check that

Table 7.2: **Complexity (number of multiplications) of evaluating and verifying linear functions.** The layers are “Fully Connected”, “Convolution”, “Depthwise Convolution” and “Pointwise Convolution”. Each layer f has an input x , output y and kernel W . We assume a batch size of $B \geq 1$. We use the shorthand $|x|$ to denote the number of elements in a vector or matrix x .

Layer	$ x , y $	$ W $	cost_f ($B = 1$)	Batch verification	With preproc.
FC	$h_{\text{in}},$ h_{out}	$h_{\text{in}} \cdot h_{\text{out}}$	$ x \cdot y $	$B \cdot (x + y) + \text{cost}_f$	$B \cdot (x + y)$
Conv	$h \cdot w \cdot c_{\text{in}},$ $h \cdot w \cdot c_{\text{out}}$	$k^2 \cdot c_{\text{in}} \cdot c_{\text{out}}$	$ x \cdot k^2 \cdot c_{\text{out}}$	$B \cdot (x + y)$ $+ c_{\text{in}} \cdot c_{\text{out}} + x \cdot k^2$	$B \cdot (x + y)$
Depth. Conv	$h \cdot w \cdot c_{\text{in}},$ $h \cdot w \cdot c_{\text{in}}$	$k^2 \cdot c_{\text{in}}$	$ x \cdot k^2$	$B \cdot (x + y) + \text{cost}_f$	$B \cdot (x + y)$
Point. Conv	$h \cdot w \cdot c_{\text{in}},$ $h \cdot w \cdot c_{\text{out}}$	$c_{\text{in}} \cdot c_{\text{out}}$	$ x \cdot c_{\text{out}}$	$B \cdot (x + y) + c_{\text{in}} \cdot c_{\text{out}}$	$B \cdot (x + y)$

$\text{Conv}(x, Ws) = ys$ (with appropriate reshaping). For a batch of inputs X , we can also apply Freivalds’ algorithm twice to reduce both W and X .

Preprocessing. We now show how to obtain an outsourcing scheme for linear layers that has optimal verification complexity (i.e., $|x| + |y|$ operations) for single-element batches and arbitrary linear operators, while at the same time *compressing* the model’s weights (a welcome property in our memory-limited TEE model).

We leverage two facts: (1) model weights are fixed at inference time, so part of Freivalds’ check can be pre-computed; (2) the TEE can keep secrets from the host \mathcal{S} , so the random values s can be re-used across layers or inputs (if we run Freivalds’ check n times with the same secret randomness, the soundness errors grows at most by a factor n). Our verification scheme with preprocessing follows from a reformulation of Lemma 7.1:

Lemma 7.3. *Let $g : \mathbb{F}^m \rightarrow \mathbb{F}^n$ be a linear operator, $g(x) := x^\top W$. Let s be uniformly random in \mathbb{S}^n , for $\mathbb{S} \subseteq \mathbb{F}$, and let $\tilde{s} := \nabla g_x(s) = Ws$. For any $x \in \mathbb{F}^m$, $y \in \mathbb{F}^n$, we have $\Pr[y^\top s = x^\top \tilde{s} \mid y \neq g(x)] \leq 1/|\mathbb{S}|$.*

The check requires $|x| + |y|$ multiplications, and storage for s and $\tilde{s} := Ws$ (of size $|x|$ and $|y|$). To save space, we can reuse the same random s for every layer. The memory footprint of a model is then equal to the size of the inputs of all its linear layers (e.g., for VGG16 the footprint is reduced from 550MB to 36MB, see Table 7.3).

7.3.3 Input Privacy

To guarantee privacy of the client’s inputs, we use precomputed blinding factors for each outsourced computation, as described in Section 7.1.3. The TEE uses a cryptographic Pseudo Random Number Generator (PRNG) to generate blinding factors. The precomputed “unblinding factors” are encrypted and stored in untrusted memory or disk. In the online phase, the TEE regenerates the blinding factors using the same PRNG seed, and uses the precomputed unblinding factors to decrypt the output of the outsourced linear layer.

This blinding process incurs several overheads: (1) the computations on the untrusted device have to be performed over \mathbb{Z}_p so we use double-precision arithmetic. (2) The trusted and untrusted processors exchange data in-between each layer, rather than at the end of a full inference pass. (3) The TEE has to efficiently load precomputed unblinding factors, which requires either a large amount of RAM, or a fast access to disk (e.g., a PCIe SSD).

Slalom’s security is given by the following results. Let λ be a *negligible* function (for any integer $c > 0$ there exists an integer N_c such that for all $x > N_c$, $|\lambda(x)| < 1/x^c$).

Theorem 7.4. *Let Slalom be the protocol from Figure 7.1 (right), where f is an n -layer neural network, and Freivalds’ algorithm is repeated k times per layer with random vectors drawn from $\mathbb{S} \subseteq \mathbb{F}$. Assume all random values are generated using a secure PRNG with security parameter λ . Then, Slalom is a secure outsourcing scheme for f between a TEE and an untrusted co-processor \mathcal{S} with privacy and t -integrity for $t = n/|\mathbb{S}|^k - \lambda(\lambda)$.*

Proof. Let $\mathbf{st} \leftarrow \text{Preproc}$ and $\text{Outsource}\langle \text{TEE}(f, x, \mathbf{st}), \mathcal{S} \rangle$ be the outsourcing scheme defined in Figure 7.1 (right). We assume that all random values sampled by the TEE are produced by a secure cryptographically secure pseudorandom number generator (PRNG) (with elements in $\mathbb{S} \subseteq \mathbb{F}$ for the integrity-check vectors s used in Freivalds’ algorithm, and in \mathbb{F} for the blinding vectors $r^{(i)}$).

We first consider integrity. Assume that the scheme is run with input $x^{(1)}$ and that the TEE outputs $y^{(n)}$. We will bound $\Pr[y^{(n)} \neq f(x^{(1)}) \mid y^{(n)} \neq \perp]$. By the security of the PRNG, we can replace the vectors s used in Freivalds’ algorithm by truly uniformly random values in $\mathbb{S} \subseteq \mathbb{F}$, via a simple hybrid argument. For the i -th linear layer, with operator $W^{(i)}$, input $x^{(i)}$ and purported output $y^{(i)}$, we then have that $y^{(i)} \neq x^{(i)}W^{(i)}$ with probability at most $1/|\mathbb{S}|^k$. By a simple union bound, we thus have that $\Pr[y^{(n)} \neq f(x^{(1)})] \leq n/|\mathbb{S}|^k - \lambda(\lambda)$. Note that this bound holds even if the same (secret) random values s are re-used across layers.

For privacy, consider the views of an adversary \mathcal{S}^* when Slalom is run with inputs x and x' . Again, by the security of the PRNG, we consider a hybrid protocol where we replace the pre-computed blinding vectors $r^{(i)}$ by truly uniformly random values in \mathbb{F} . In this hybrid protocol, $\tilde{x}^{(i)} = x^{(i)} + r^{(i)}$ is simply a “one-time-pad” encryption of $x^{(i)}$ over the field \mathbb{F} , so \mathcal{S}^* ’s views in both executions of the hybrid protocol are equal (information theoretically). Thus, \mathcal{S}^* ’s views in both executions of the original protocol are computationally indistinguishable. \square

Corollary 7.5. *Assuming the TEE is secure (i.e., it acts as a trusted third party hosted by \mathcal{S}), Slalom is a secure outsourcing scheme between a remote client \mathcal{C} and server \mathcal{S} with privacy and t -integrity for $t = n/|\mathbb{S}|^k - \lambda(\lambda)$. If the model f is the property of \mathcal{S} , the scheme further satisfies model privacy.*

Proof. The outsourcing protocol between the remote client \mathcal{C} and server \mathcal{S} hosting the TEE is simply defined as follows (we assume the model belongs to \mathcal{S}):

- $\text{st} \leftarrow \text{Preproc}()$: \mathcal{C} and the TEE setup a secure authenticated communication channel, using the TEE’s remote attestation property. The TEE receives the model f from \mathcal{S} and initializes the Slalom protocol.
- $\text{Outsource}(\mathcal{C}(x, \text{st}), \mathcal{S}(f))$:
 - \mathcal{C} sends x to the TEE over the secure channel.
 - The TEE securely computes $y = f(x)$ using Slalom.
 - The TEE sends y (and a publicly verifiable commitment to f) to \mathcal{C} over the secure channel.

If the TEE is secure (i.e., it acts as a trusted third party hosted by \mathcal{S}), then the result follows. \square

7.4 Empirical Evaluation

We evaluate Slalom on real Intel SGX hardware, on micro-benchmarks and a sample application (ImageNet inference with VGG16, MobileNet and ResNet models). Our aim is to show that, compared to a baseline that runs inference fully in the TEE, outsourcing linear layers increases performance without sacrificing security.

7.4.1 Implementation

As enclaves cannot access most OS features (e.g., multi-threading, disk and driver IO), porting a large framework such as TensorFlow or Intel’s MKL-DNN to SGX is hard. Instead, we designed a lightweight C++ library for feed-forward networks based on Eigen, a linear-algebra library which TensorFlow uses as a CPU backend. Our library implements the forward pass of neural networks, with support for dense layers, standard and separable convolutions, pooling, and activations. When run on a native CPU (without SGX), its performance is comparable to TensorFlow on CPU (compiled with AVX).

Slalom performs arithmetic over \mathbb{Z}_p , for $p = 2^{24} - 3$. For integrity, we apply Freivalds’ check twice to each layer ($k = 2$), with random values from $\mathbb{S} = [-2^{19}, 2^{19}]$, to achieve 40 bits of *statistical soundness* per layer (see Section 7.3.1 for details on the selection of these parameters). For a 50-layer network, \mathcal{S} has a chance of less than 1 in 22 billion of fooling the TEE on any incorrect model

Table 7.3: **Details of models used to evaluate Slalom.** Accuracies are computed on the ImageNet validation set. Pre-trained models are from Keras [46].

Model	Accuracy		Quantized		Layers	Params (M)	Layer in/out (M)
	Top 1	Top 5	Top 1	Top 5			
VGG16	71.0	90.0	70.6	89.5	16	138.4	9.1 / 13.6
VGG16 (no top)	-	-	-	-	13	14.7	9.1 / 13.5
MobileNet	70.7	89.6	70.5	89.5	28	4.2	5.5 / 5.0
MobileNet (fused)	-	-	-	-	15	4.2	3.6 / 3.1
ResNet 50	76.9	92.4	76.4	92.2	50	25.5	10.0 / 10.4

evaluation (a slightly better guarantee than in SafetyNets). For privacy, we use AES-CTR and AES-GCM to generate, encrypt and authenticate blinding factors.

7.4.2 Setup

We use an Intel Core i7-6700 Skylake 3.40GHz processor with 8GB of RAM, a desktop processor with SGX support. The outsourced computations are performed on a co-located Nvidia TITAN XP GPU. Due to a lack of native internal multi-threading in SGX, we run our TEE in a single CPU thread. We discuss challenges for efficient parallelization in Section 7.4.6. We evaluate Slalom on the following workloads:

- Synthetic benchmarks for matrix products, convolutions and separable convolutions, where we compare the enclave’s running time for computing a linear operation to that of solely verifying the result.
- ImageNet [61] classification with VGG16 [236], MobileNet [109], and ResNet [105] models (with fused Batch Normalization layers when applicable).

MobileNet, a model tailored for low compute devices, serves as a worst-case benchmark for Slalom, as the model’s design aggressively minimizes the amount of computation performed per layer. We also consider a “fused” variant of MobileNet with no activation between depthwise and pointwise convolutions. Removing these activations *improves* convergence and accuracy [47, 231], while also making the network more outsourcing-friendly (i.e., it is possible to verify a separable convolution in a single step).

Our evaluation focuses on throughput (number of forward passes per second). We also discuss energy efficiency to account for hardware differences between our baseline (TEE only) and Slalom (TEE + GPU), see section 7.4.4, A note on energy efficiency.

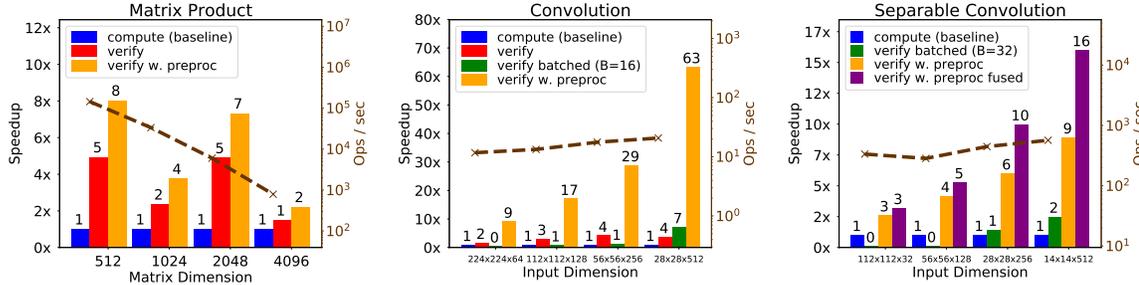


Figure 7.2: **Micro benchmarks on Intel SGX.** We plot the relative speedup of verifying the result of a linear operator compared to computing it entirely in the enclave. The dotted line shows the throughput obtained for a direct computation. “Fused” separable convolutions contain no intermediate activation.

7.4.3 Neural Network Details

Table 7.3 provides details about the two models we use in our evaluation (all pre-trained models are taken from Keras [46]). We report top 1 and top 5 accuracy on ImageNet with and without the simple quantization scheme described in Section 7.3.1. Quantization results in at most a 0.5% drop in top 1 and top 5 accuracy. More elaborate quantization schemes exist (e.g., Micikevicius et al. [169]) that we have not experimented with in this work.

We report the number of model parameters, which is relevant to the memory constraints of TEEs such as Intel SGX. We also list the total size of the inputs and outputs of all the model’s linear layers, which impact the amount of communication between trusted and untrusted co-processors in Slalom, as well as the amount of data stored in the TEE when using Freivalds’ algorithm with preprocessing.

7.4.4 Results

Micro-benchmarks. Our micro-benchmark suite consists of square matrix products of increasing dimensions, convolutional operations performed by VGG16, and separable convolutions performed by MobileNet. In all cases, the data is pre-loaded inside an enclave, so we only measure the in-enclave execution time. Figure 7.2 plots the relative speedups of various verification strategies over the cost of computing the linear operation directly. In all cases, the baseline computation is performed in single-precision floating point, and the verification algorithms repeat Freivalds’ check so as to attain at least 40 bits of statistical soundness.

For square matrices of dimensions up to 2048, verifying an outsourced result is 4× to 8× faster than computing it. For larger matrices, we exceed the limit of SGX’s DRAM, so the enclave resorts to expensive paging which drastically reduces performance both for computation and verification.

For convolutions (standard or separable), we achieve large savings with outsourcing if Freivalds’ algorithm is applied with preprocessing. The savings get higher as the number of channels increases.

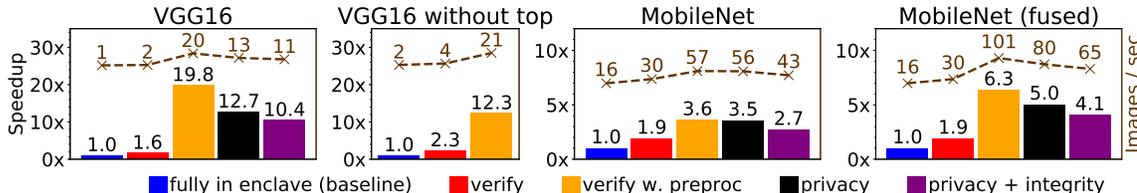


Figure 7.3: **Verifiable and private inference with Intel SGX.** We show results for VGG16, VGG16 without the fully connected layers, MobileNet, and a fused MobileNet variant with no intermediate activation for separable convolutions. We compare the baseline of fully executing the model in the enclave (blue) to different secure outsourcing schemes: integrity with Freivalds (red); integrity with Freivalds and precomputed secrets (yellow); privacy only (black); privacy and integrity (purple).

Without preprocessing, Freivalds’ algorithm results in savings when c_{out} is large. Due to SGX’s small PRM, batched verification is only effective for operators with small memory footprints. As expected, “truly” separable convolutions (with no intermediate non-linearity) are much faster to verify, as they can be viewed as a single linear operator.

Verifiable inference. Figure 7.3 shows the throughput of end-to-end forward passes in two neural networks, VGG16 and MobileNet. For integrity, we compare the secure baseline (executing the model fully in the enclave) to two variants of the Slalom algorithm in Figure 7.1. The first (in red) applies Freivalds’ algorithm “on-the-fly”, while the second more efficient variant (in orange) pre-computes part of Freivalds’ check as described in Section 7.3.2.

The VGG16 network is much larger (500MB) than SGX’s PRM. As a result, there is a large overhead on the forward pass and verification without preprocessing. If the enclave securely stores preprocessed products Wr for all network weights, we drastically reduce the memory footprint and achieve up to a $20.3\times$ increase in throughput. We also ran the lower-half of the VGG16 network (without the fully connected layers), a common approach for extracting features for transfer learning or object recognition [153]. This part fits in the PRM, and we thus achieve higher throughput for in-enclave forward passes and on-the-fly verification.

For MobileNet, we achieve between $3.6\times$ and $6.4\times$ speedups when using Slalom for verifiable inference (for the standard or “fused” model, respectively). The speedups are smaller than for VGG16, as MobileNet performs much fewer operations per layer (verifying a linear layer requires computing at least two multiplications for each input and output. The closer the forward pass gets to that lower-bound, the less we can save by outsourcing).

Private inference. We further benchmark the cost of private neural network inference, where inputs of outsourced linear layers are additionally *blinded*. Blinding and unblinding each layer’s inputs and outputs is costly, especially in SGX due to the extra in-enclave memory reads and

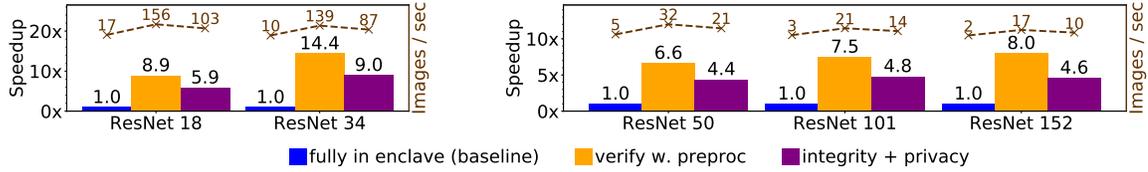


Figure 7.4: **Secure outsourcing of ResNet models with Intel SGX.** We compare the baseline of fully executing the model in the enclave (blue) to secure outsourcing with integrity (yellow) and privacy and integrity (purple).

writes. Nevertheless, for VGG16 and the fused MobileNet variant without intermediate activations, we achieve respective speedups of 13.0 \times and 5.0 \times for private outsourcing (in black in Figure 7.3), and speedups of 10.7 \times and 4.1 \times when also ensuring integrity (in purple). For this benchmark, the precomputed unblinding factors are stored in untrusted memory.

We performed the same experiments on a standard CPU (i.e., without SGX) and find that Slalom’s improvements are even higher in non-resource-constrained or multi-threaded environments (see Section 7.4.5 and Section 7.4.6). Slalom’s improvements over the baseline also hold when accounting for energy efficiency (see section 7.4.4, A note on energy efficiency below).

Extending Slalom to deep residual networks. The Slalom algorithm in Figure 7.1 and our evaluations above focus on feed-forward architectures. Extending Slalom to more complex neural networks is quite simple. To illustrate, we consider the family of ResNet models [105], which use residual blocks $g(x) = \sigma(g_1(x) + g_2(x))$ that merge two feed-forward “paths” g_1 and g_2 into a final activation σ . To verify integrity of $g(x)$, the TEE simply verifies all linear layers in g_1 and g_2 and computes σ directly. For privacy, the TEE applies the interactive Slalom protocol in Figure 7.1 (right) in turn to g_1 and g_2 , and then computes σ . The results for the *privacy-preserving* Slalom variant in Figure 7.4 use a preliminary implementation that performs all required operations—and thus provides meaningful performance numbers—but without properly constructed unblinding factors.

We use the ResNet implementation from Keras [46], which contains a pre-trained 50-layer variant. For this model, we find that our quantization scheme results in less than a 0.5% decrease in accuracy (see Table 7.3). For other variants (i.e., with 18, 34, 101 and 152 layers) we compute throughput on untrained models. Figure 7.4 shows benchmarks for different ResNet variants when executed fully in the enclave (our baseline) as well as secure outsourcing with integrity or privacy and integrity. For all models, we achieve 6.6 \times to 14.4 \times speedups for verifiable inference and 4.4 \times to 9.0 \times speedups when adding privacy.

Comparing results for different models is illustrative of how Slalom’s savings scale with model size and architectural design choices. The 18 and 34-layer ResNets use convolutions with 3×3 kernels, whereas the larger models mainly use pointwise convolutions. As shown in Table 7.2 verifying a

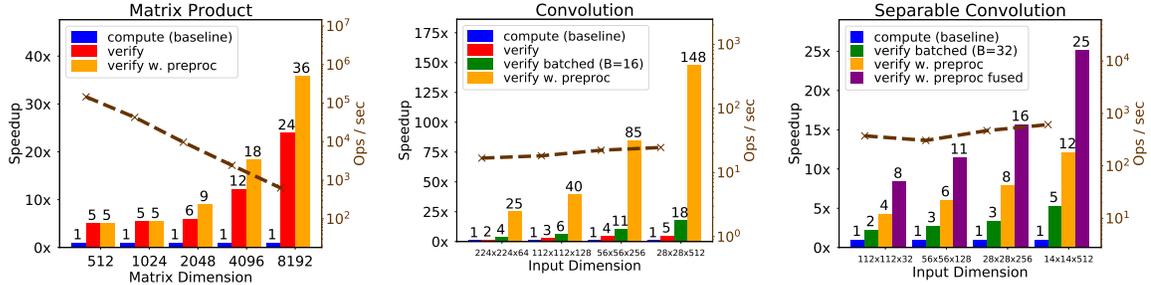


Figure 7.5: **Micro benchmarks on an untrusted CPU.** For three different linear operators, we plot the relative speedup of verifying a result compared to computing it. The dotted line in each plot shows the throughput obtained for computing the operation.

convolution is about a factor $k^2 \cdot c_{\text{out}}$ than computing it, which explains the higher savings for models that use convolutions with large kernel windows. When adding more layers to a model, we expect Slalom’s speedup over the baseline to remain constant (e.g., if we duplicate each layer, the baseline computation and the verification should both take twice as long). Yet we find that Slalom’s speedups usually *increase* as layers get added to the ResNet architecture. This is because the deeper ResNet variants are obtained by duplicating layers towards the end of the pipeline, which have the largest number of channels and for which Slalom achieves the highest savings.

A note on energy efficiency. When comparing approaches with different hardware (e.g., our single-core CPU baseline versus Slalom which also uses a GPU), throughput alone is not the fairest metric. E.g., the baseline’s throughput could also be increased by adding more SGX CPUs. A more accurate comparison considers the *energy efficiency* of a particular approach, a more direct measure of the recurrent costs to the server \mathcal{S} .

For example, when evaluating MobileNet or VGG16, our GPU draws 85W of power, whereas our baseline SGX CPU draws 30W. As noted above, the GPU also achieves more than 50 \times higher throughput, and thus is at least 18 \times more energy efficient (e.g., measured in Joules per image) than the enclave.

For Slalom, we must consider the cost of running both the enclave and GPU. In our evaluations, the outsourced computations on the GPU account for at most 10% of the total running time of Slalom (i.e., the integrity checks and data encryption/decryption in the enclave are the main bottleneck). Thus, the power consumption attributed to Slalom is roughly $10\% \cdot 85\text{W} + 90\% \cdot 30\text{W} = 35.5\text{W}$. Note that when not being in use by Slalom, the trusted CPU or untrusted GPU can be used by other tasks running on the server. As Slalom achieves 4 \times -20 \times higher throughput than our baseline for the tasks we evaluate, it is also about 3.4 \times -17.1 \times more energy efficient.

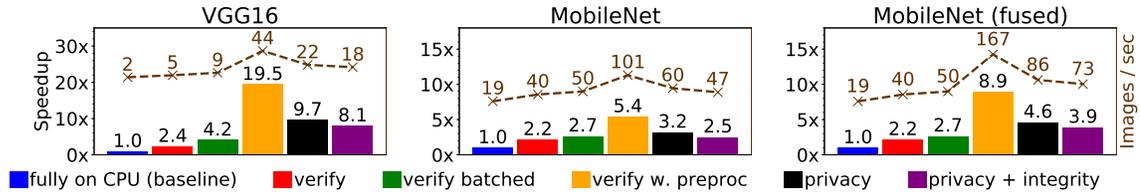


Figure 7.6: **Inference with integrity and privacy on an untrusted CPU.** We compare the baseline inference throughput (blue) to that obtained with “on-the-fly” integrity checks (red); batched integrity checks (green); integrity checks with precomputed secrets (yellow); privacy only (black); and privacy and integrity (purple). The fused MobileNet model has no intermediate activation for separable convolutions.

7.4.5 Results on a Standard CPU

For completeness, and to assess how our outsourcing scheme fairs in an environment devoid of Intel SGX’s performance quirks, we rerun the evaluations in Section 7.4 on the same CPU but outside of SGX’s enclave mode.

Figure 7.5 shows the results of the micro-benchmarks for matrix multiplication, convolution and separable convolutions. In all cases, verifying a computation becomes 1-2 orders of magnitude faster than computing it as the outer dimension grows. Compared to the SGX benchmarks, we also see a much better viability of batched verification (we haven’t optimized batched verifications much, as they are inherently slow on SGX. It is likely that these numbers could be improved significantly, to approach those of verification with preprocessing).

Figure 7.6 shows benchmarks for VGG16 and MobileNet on a single core with either direct computation or various secure outsourcing strategies. For integrity alone, we achieve savings up to 8.9× and 19.5× for MobileNet and VGG16 respectively. Even without storing any secrets in the enclave, we obtain good speedups using batched verification. As noted above, it is likely that the batched results could be further improved. With additional blinding to preserve privacy, we achieve speedups of 3.9× and 8.1× for MobileNet and VGG16 respectively.

7.4.6 Parallelization

Our experiments on SGX in Section 7.4 were performed using a single execution thread, as SGX enclaves do not have the ability to create threads. We have also experimented with techniques for achieving parallelism in SGX, both for standard computations and outsourced ones, but with little success.

To optimize for throughput, a simple approach is to run multiple forward passes simultaneously. On a standard CPU, this form of “outer-parallelism” achieves close to linear scaling as we increase the number of threads from 1 to 4 on our quad-core machine. With SGX however, we did not manage to achieve any parallel speedup for VGG16—whether for direct computation or verifying outsourced

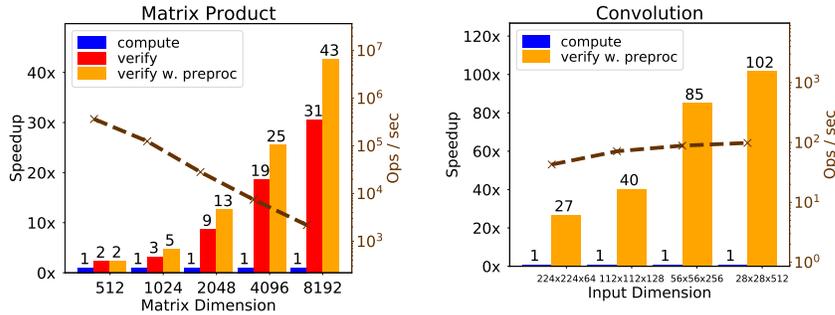


Figure 7.7: **Multi-threaded micro benchmarks on an untrusted CPU.** Reiterates benchmarks for matrix products and convolutions using 4 threads.

results—presumably because each independent thread requires extra memory that quickly exceeds the PRM limit. For the smaller MobileNet model, we get less than a $1.5\times$ speedup using up to 4 threads, for direct computation or outsourced verification alike.

Neural networks typically also make use of intra-operation parallelism, i.e., computing the output of a given layer using multiple threads. Our neural network library currently does not support intra-operation parallelism, but implementing a dedicated thread pool for SGX could be an interesting extension for future work. Instead, we evaluate the potential benefits of intra-op parallelism on a standard untrusted CPU, for our matrix-product and convolution benchmarks. We make use of Eigen’s internal multi-threading support to speed up these operations, and custom OpenMP code to parallelize dot products, as Eigen does not do this on its own.

Figure 7.7 shows the results using 4 threads. For convolutions, we have currently only implemented multi-threading for the verification with preprocessing (which requires only standard dot products). Surprisingly maybe, we find that multi-threading *increases* the gap between direct and verified computations of matrix products, probably because dot products are extremely easy to parallelize efficiently (compared to full convolutions). We also obtain close to linear speedups for verifiable separable convolutions, but omit the results as we currently do not have an implementation of multi-threaded direct computation for depthwise convolutions, which renders the comparison unfair. Due to the various memory-access overheads in SGX, it is unclear whether similar speedups could be obtained by using intra-op parallelism in an enclave, but this is an avenue worth exploring.

7.5 Challenges for Verifiable and Private Training

Our techniques for secure outsourcing of neural network inference might also apply to model training. Indeed, a backward pass consists of similar linear operators as a forward pass, and can thus be verified with Freivalds’ algorithm. Yet, applying Slalom to neural network training is challenging, as described below, and we leave this problem open.

- Quantizing neural networks for training is harder than for inference, due to large changes in weight magnitudes [169]. Thus, a more flexible quantization scheme than the one we used would be necessary.
- Because the model’s weights change during training, the same preprocessed random vectors for Freivalds’ check cannot be re-used indefinitely. The most efficient approach would presumably be to train with very large batches that can then be verified simultaneously.
- Finally, the pre-computation techniques we employ for protecting input privacy do not apply for training, as the weights change after every processed batch. Moreover, Slalom does not try to hide the model weights from the untrusted processor, which might be a requirement for private training.

7.6 Conclusion

We have studied the efficiency of evaluating a neural network in a Trusted Execution Environment (TEE) to provide strong integrity and privacy guarantees. We explored new approaches for segmenting a neural network evaluation to securely outsource work from a trusted environment to a faster co-located but untrusted processor.

We designed Slalom, a framework for efficient neural network evaluation that outsources all linear layers from a TEE to a GPU. Slalom leverages Freivalds’ algorithm for verifying correctness of linear operators, and additionally encrypts inputs with precomputed blinding factors to preserve privacy. Slalom can work with any TEE and we evaluated its performance using Intel SGX on various workloads. For canonical image classifiers (VGG16, MobileNet and ResNet variants), we have shown that Slalom boosts inference throughput without compromising security.

Securely outsourcing matrix products from a TEE has applications in ML beyond neural network (e.g., non negative matrix factorization, dimensionality reduction, etc.) We have also explored avenues and challenges towards applying similar techniques to neural network training, an interesting direction for future work. Finally, our general approach of outsourcing work from a TEE to a faster co-processor could be applied to other problems which have fast verification algorithms, e.g., those considered in [163, 287].

Part III

Conclusion

Despite substantial recent improvements in the predictive capabilities of machine learning systems, these systems remain surprisingly prone to mistakes that threaten the security and privacy of their users. To fully reap the benefits of data-driven systems in security- or safety-critical applications, it is thus imperative that we better understand and ultimately mitigate the flaws of current machine learning models.

In this dissertation, we have introduced new approaches and techniques for measuring and enhancing the integrity and privacy of machine learning. We have critically assessed the threat posed by adversarial examples to existing machine learning applications, and demonstrated a more compelling use-case in the context of online content blocking. We have further shown that existing defense techniques only cover an overly simplistic threat model, and are inherently limited against realistic attacks. To enhance the privacy of machine learning users, we have developed new techniques for private training and inference that significantly reduce the utility gap compared to non-private baselines.

The results in this dissertation pave the way towards a more rigorous assessment of the security risks faced by machine learning models deployed in adversarial settings, and present encouraging avenues towards building high-performing learning systems that refrain from needlessly endangering users' privacy.

Bibliography

- [1] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.
- [2] Tiago Alves and Don Felton. Trustzone: Integrated hardware and software security-enabling trusted computing in embedded systems. Technical report, ARM, 2004.
- [3] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.
- [4] Mathieu Andreux, Tomás Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Joakim Andén, Eugene Belilovsky, Joan Bruna, Vincent Lostanlen, Matthew J. Hirn, Edouard Oyallon, Sixin Zhang, Carmine Cella, and Michael Eickenberg. Kymatio: Scattering transforms in Python. *Journal of Machine Learning Research*, 21(60):1–6, 2020.
- [5] Sanjeev Arora, Simon S Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020.
- [6] 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*, 2018.
- [7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, 2018.
- [8] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on Graphics*, volume 26, page 10, 2007.
- [9] Mitali Bafna, Jack Murtagh, and Nikhil Vyas. Thwarting adversarial examples: An ℓ_0 -robust sparse Fourier transform. In *Advances in Neural Information Processing Systems*, pages 10096–10106, 2018.

- [10] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, pages 15479–15488, 2019.
- [11] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *IEEE Symposium on Foundations of Computer Science*, pages 464–473, 2014.
- [12] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *ACM Symposium on Theory of Computing*, pages 1–10, 1988.
- [13] Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, and Cherita L Corbett. A survey of deep learning methods for cyber security. *Information*, 10(4):122, 2019.
- [14] Andrew C Berry. The accuracy of the Gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136, 1941.
- [15] Sruti Bhagavatula, Christopher Dunn, Chris Kanich, Minaxi Gupta, and Brian Ziebart. Leveraging machine learning to improve unwanted resource filtering. In *ACM Workshop on Artificial Intelligence and Security*, 2014.
- [16] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.
- [17] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.
- [18] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [19] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.
- [20] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiaainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. In *USENIX Workshop on Offensive Technologies*, 2017.

- [21] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.
- [22] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.
- [23] Zhiqi Bu, Jinshuo Dong, Qi Long, and Su Weijie. Deep learning with Gaussian differential privacy. *Harvard Data Science Review*, 9 2020.
- [24] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C Mitchell. The end is nigh: Generic solving of text-based CAPTCHAs. In *USENIX Workshop on Offensive Technologies*, 2014.
- [25] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *ACM Symposium on Theory of Computing*, pages 494–503, 2002.
- [26] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.
- [27] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.
- [28] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Workshop on Deep Learning Security*, 2018.
- [29] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, pages 513–530, 2016.
- [30] Nicholas Carlini, Úlfar Erlingsson, and Nicolas Papernot. Prototypical examples in deep learning: Metrics, characteristics, and utility, 2019. URL <https://openreview.net/forum?id=r1xyx3R9tQ>.
- [31] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, pages 267–284, 2019.
- [32] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

- [33] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11192–11203, 2019.
- [34] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.
- [35] Chen Chen and Jaewoo Lee. Stochastic adaptive line search for differentially private optimization. In *IEEE International Conference on Big Data*, 2020.
- [36] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. SGXPECTRE: Stealing intel secrets from SGX enclaves via speculative execution. In *IEEE European Symposium on Security and Privacy*, 2019.
- [37] Jianbo Chen and Michael I Jordan. HopSkipJumpAttack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy*, 2020.
- [38] Mia Xu Chen, Benjamin N. Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. Gmail smart compose: Real-time assisted writing. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2287–2295, 2019.
- [39] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. EAD: elastic-net attacks to deep neural networks via adversarial examples. In *AAAI Conference on Artificial Intelligence*, 2018.
- [40] Sanchuan Chen, Xiaokuan Zhang, Michael K Reiter, and Yinqian Zhang. Detecting privileged side-channel attacks in shielded execution with Déjà Vu. In *ACM Asia Conference on Computer and Communications Security*, pages 7–18, 2017.
- [41] Steven Chen, Nicholas Carlini, and David Wagner. Stateful detection of black-box adversarial attacks. In *ACM Workshop on Security and Privacy on Artificial Intelligence*, 2020.
- [42] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.
- [43] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, 2020.
- [44] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

- [45] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. In *IEEE European Symposium on Security and Privacy*, 2019.
- [46] François Chollet et al. Keras. <https://keras.io>, 2015.
- [47] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [48] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. Sentinet: Detecting physical attacks against deep learning systems. In *IEEE Workshop on Deep Learning Security*, 2020.
- [49] Nicolas Christin, Sally S Yanagihara, and Keisuke Kamataki. Dissecting one click frauds. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 15–26, 2010.
- [50] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. In *Advances in Neural Information Processing Systems*, 2017.
- [51] Kenneth T Co, Luis Muñoz-González, Sixte de Maupéou, and Emil C Lupu. Procedural noise adversarial examples for black-box attacks on deep convolutional networks. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.
- [52] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012.
- [53] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.
- [54] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.
- [55] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, 2016 (86), 2016. URL <http://eprint.iacr.org/2016/086>.
- [56] Victor Costan, Ilija Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *USENIX Security Symposium*, 2016.
- [57] Justin Crites and Mathias Ricken. Automatic ad blocking: Improving Adblock for the Mozilla platform. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.544.5305>.
- [58] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

- [59] Fergus Dall, Gabrielle De Micheli, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi, and Yuval Yarom. Cachequote: Efficiently recovering long-term secrets of SGX EPID via cache attacks. *International Conference on Cryptographic Hardware and Embedded Systems*, 2018(2):171–191, 2018.
- [60] Ambra Demontis, Paolo Russu, Battista Biggio, Giorgio Fumera, and Fabio Roli. On security and sparsity of linear classifiers for adversarial settings. In *IAPR Workshop on Statistical Techniques in Pattern Recognition*, pages 322–332. Springer, 2016.
- [61] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [62] Digital Advertising Alliance (DAA). Self regulatory principles for online behavioral advertising. https://digitaladvertisingalliance.org/sites/aboutads/files/DAA_files/seven-principles-07-01-09.pdf, 2009. Accessed: 2021-06-22.
- [63] Digital Advertising Alliance (DAA). DAA icon ad marker creative guidelines. https://digitaladvertisingalliance.org/sites/aboutads/files/DAA_files/DAA_Icon_Ad_Creative_Guidelines.pdf, 2013. Accessed: 2021-06-22.
- [64] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l_1 -ball for learning in high dimensions. In *International Conference on Machine Learning*, 2008.
- [65] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT*, pages 486–503. Springer, 2006.
- [66] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.
- [67] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *ACM Symposium on Theory of Computing*, pages 117–126, 2015.
- [68] The Economist. The world’s most valuable resource is no longer oil, but data. *The Economist*, 2017.
- [69] Benjamin Edelman. False and deceptive display ads at Yahoo’s right media. <http://www.benedelman.org/rightmedia-deception/#reg>, 2009. Accessed: 2021-06-22.

- [70] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.
- [71] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, 2019.
- [72] Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 37(2):505–515, 2017.
- [73] 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 *USENIX Workshop on Offensive Technologies*, 2018.
- [74] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.
- [75] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1186–1195, 2018.
- [76] Vitaly Feldman. Does learning require memorization? A short tale about a long tail. In *ACM Symposium on Theory of Computing*, pages 954–959, 2020.
- [77] Vitaly Feldman and Tijana Zrnic. Individual privacy accounting via a Rényi filter. *arXiv preprint arXiv:2008.11193*, 2020.
- [78] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *IEEE Symposium on Foundations of Computer Science*, pages 521–532, 2018.
- [79] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 501–512, 2012.
- [80] Ben Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. Iron: functional encryption using Intel SGX. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 765–782, 2017.
- [81] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

- [82] Rusins Freivalds. Probabilistic machines can use less running time. In *IFIP Congress on Information Processing*, volume 839, page 842, 1977.
- [83] Timon Gehr, Matthew Mirman, Dana Drachler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy*, 2018.
- [84] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.
- [85] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing*, pages 169–178, 2009.
- [86] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.
- [87] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. In *Advances in Neural Information Processing Systems*, pages 4675–4684, 2017.
- [88] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.
- [89] Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.
- [90] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.
- [91] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *ACM Symposium on Theory of Computing*, pages 218–229, 1987.
- [92] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
- [93] Ian Goodfellow. A research agenda: Dynamic models to defend against correlated attacks. *arXiv preprint arXiv:1903.06293*, 2019.
- [94] Ian Goodfellow and Nicolas Papernot. Is attacking machine learning easier than defending it? *Blog post on Feb*, 15:2017, 2017.

- [95] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- [96] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on Intel SGX. In *ACM European Workshop on Systems Security*, page 2, 2017.
- [97] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
- [98] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.
- [99] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, 2017.
- [100] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.
- [101] David Gugelmann, Markus Happe, Bernhard Ager, and Vincent Lenders. An automated approach for complementing ad blockers’ blacklists. *Privacy Enhancing Technologies*, 2:282–298, 2015.
- [102] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.
- [103] Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Max Augustin, Michael Backes, and Mario Fritz. MLCapsule: Guarded offline deployment of machine learning as a service. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2021.
- [104] Danny Harnik and Eliad Tsfadia. Impressions of Intel SGX performance. https://medium.com/@danny_harnik/impressions-of-intel-sgx-performance-22442093595a, 2017. Accessed: 2021-06-22.
- [105] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [106] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. In *USENIX Workshop on Offensive Technologies*, 2017.

- [107] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.
- [108] Giovanni Hernandez, Akshay Jagadeesh, and Jonathan Mayer. Tracking the trackers: The AdChoices icon. <http://cyberlaw.stanford.edu/blog/2011/08/tracking-trackers-adchoices-icon>, 2011. Accessed: 2021-06-22.
- [109] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [110] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. Secure and robust SIFT. In *ACM International Conference on Multimedia*, pages 637–640, 2009.
- [111] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.
- [112] Zaeem Hussain, Mingda Zhang, Xiaozhong Zhang, Keren Ye, Christopher Thomas, Zuha Agha, Nathan Ong, and Adriana Kovashka. Automatic understanding of image and video advertisements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1100–1110, 2017.
- [113] Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.
- [114] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018.
- [115] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, 2019.
- [116] Intel Corp. Intel Software Guard Extensions Evaluation SDK. <https://software.intel.com/en-us/sgx-sdk>, 2015. Accessed: 2021-06-22.
- [117] Intel Corp. Intel software guard extensions (SGX) SW development guidance for potential bounds check bypass (CVE-2017-5753) side channel exploits. https://software.intel.com/sites/default/files/180204_SGX_SDK_Developer_Guidance_v1.0.pdf, 2018. Accessed: 2021-06-22.
- [118] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *ACM Internet Measurement Conference*, pages 171–183, 2017.

- [119] Umar Iqbal, Zubair Shafiq, Peter Snyder, Shitong Zhu, Zhiyun Qian, and Benjamin Livshits. AdGraph: A machine learning approach to automatic and effective adblocking. In *IEEE Symposium on Security and Privacy*, 2020.
- [120] Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *International Conference on Learning Representations*, 2019.
- [121] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pages 8571–8580, 2018.
- [122] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private SGD? In *Advances in Neural Information Processing Systems*, 2020.
- [123] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems*, pages 6343–6354, 2018.
- [124] Jason Jo and Yoshua Bengio. Measuring the tendency of CNNs to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.
- [125] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *International Symposium on Computer Architecture*, pages 1–12, 2017.
- [126] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. Gazelle: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, 2018.
- [127] Peter Kairouz, Mónica Ribero, Keith Rush, and Abhradeep Thakurta. Fast dimension independent private AdaGrad on publicly estimated subspaces. *arXiv preprint arXiv:2008.06570*, 2020.
- [128] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D’Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer

- Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1), 2021.
- [129] Daniel Kang, Yi Sun, Tom Brown, Dan Hendrycks, and Jacob Steinhardt. Transfer of adversarial robustness between perturbation types. *arXiv preprint arXiv:1905.01034*, 2019.
- [130] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.
- [131] Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020.
- [132] Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. *arXiv preprint arXiv:1811.00525*, 2018.
- [133] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, pages 25–1, 2012.
- [134] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [135] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *IEEE Symposium on Security and Privacy*, 2019.
- [136] Ilker Koksall. How Alexa is changing the future of advertising. <https://www.forbes.com/sites/ilkerkoksall/2018/12/11/how-alexa-is-changing-the-future-of-advertising>, 2018. Accessed: 2021-06-22.
- [137] Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 2017.
- [138] Georgios Kontaxis and Monica Chew. Tracking protection in Firefox for privacy and performance. In *Web 2.0 Security and Privacy Workshop*, 2015.
- [139] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better ImageNet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

- [140] Viktor Krammer. An effective defense against intrusive web advertising. In *IEEE Conference on Privacy, Security and Trust*, pages 3–14, 2008.
- [141] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.
- [142] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [143] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations*, 2017.
- [144] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.
- [145] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST handwritten digit database. *ATT Labs*, 2010.
- [146] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy*, pages 656–672, 2019.
- [147] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In *USENIX Security Symposium*, pages 16–18, 2017.
- [148] Pedro Giovanni Leon, Justin Cranshaw, Lorrie Faith Cranor, Jim Graves, Manoj Hastak, Blase Ur, and Guzi Xu. What do online behavioral advertising privacy disclosures communicate to users? In *ACM Workshop on Privacy in the electronic society*, pages 19–30, 2012.
- [149] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.
- [150] Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.
- [151] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *ACM SIGSAC Conference on Computer and Communications Security*, 2012.
- [152] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *ACM Symposium on Theory of Computing*, pages 298–309, 2019.

- [153] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.
- [154] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.
- [155] David G Lowe. Object recognition from local scale-invariant features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1150–1157, 1999.
- [156] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [157] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on MRI. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.
- [158] Aleksander Madry and Zico Kolter. Adversarial robustness: Theory and practice. In *Tutorial at NeurIPS 2018*, 2018. URL <https://adversarial-ml-tutorial.org/>. Accessed: 2021-06-22.
- [159] 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.
- [160] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *AAAI Conference on Artificial Intelligence*, 2019.
- [161] Matthew Malloy, Mark McNamara, Aaron Cahn, and Paul Barford. Ad blockers: Global prevalence and impact. In *ACM Internet Measurement Conference*, pages 119–125, 2016.
- [162] Christopher Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001.
- [163] Ross M McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.
- [164] Frank McKeen, Ilya Alex, Alex Berenzon, Carlos Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday Savagaonkar. Innovative instructions and software model for isolated execution. In *ACM Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

- [165] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.
- [166] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.
- [167] Jeremy B. Merrill and Ariana Tobin. Facebook moves to block ad transparency tools - including ours. <https://www.propublica.org/article/facebook-blocks-ad-transparency-tools>, 2019. Accessed: 2021-06-22.
- [168] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.
- [169] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.
- [170] Ilya Mironov. Rényi differential privacy. In *IEEE Computer Security Foundations Symposium*, pages 263–275, 2017.
- [171] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled Gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.
- [172] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [173] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. Cachezoom: How SGX amplifies the power of cache attacks. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 69–90. Springer, 2017.
- [174] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pages 19–38, 2017.
- [175] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1765–1773, 2017.
- [176] Muhammad Haris Mughees, Zhiyun Qian, Zubair Shafiq, Karishma Dash, and Pan Hui. A first look at ad-block detection: A new arms race on the Web. *arXiv preprint arXiv:1605.05841*, 2016.

- [177] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. Detecting anti ad-blockers in the wild. In *Privacy Enhancing Technologies*, volume 3, pages 130–146, 2017.
- [178] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.
- [179] Milad Nasr, Reza Shokri, and Amir Houmansadr. Improving deep learning with differential privacy using gradient encoding and denoising. *arXiv preprint arXiv:2007.11524*, 2020.
- [180] Meghan Neal. You’re going to need an ad blocker for your next TV. https://motherboard.vice.com/en_us/article/mg7ek8/youre-going-to-need-an-ad-blocker-for-your-next-tv, 2016. Accessed: 2021-06-22.
- [181] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *ACM Symposium on Theory of Computing*, pages 75–84, 2007.
- [182] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E Powles, ED Cristofaro, Hamed Haddadi, and Steven J Murdoch. Ad-blocking and counter blocking: A slice of the arms race. In *USENIX Workshop on Free and Open Communications on the Internet*, 2016.
- [183] Olga Ohrimenko, Felix Schuster, Cdric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security Symposium*, 2016.
- [184] Paraska Oleksandr. Towards more intelligent ad blocking on the web. <https://medium.com/@shoniko/towards-more-intelligent-ad-blocking-on-the-web-9f67bf2a12b5>, 2018. Accessed: 2021-06-22.
- [185] Catherine Olsson. Unsolved research problems vs. real-world threat models. <https://medium.com/@catherio/unsolved-research-problems-vs-real-world-threat-models-e270e256bc9e>, Mar 2019. Accessed: 2021-06-22.
- [186] Meni Orenbach, Pavel Lifshits, Marina Minkin, and Mark Silberstein. Eleos: Exitless OS services for SGX enclaves. In *ACM European Conference on Computer Systems*, pages 238–253, 2017.
- [187] Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2873, 2015.
- [188] Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, and Eugene Belilovsky. Scattering networks for hybrid representation

- learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2208–2221, 2018.
- [189] George Paliy. The future of advertising in virtual reality. <https://www.kivodaily.com/technology/the-future-of-advertising-in-virtual-reality/>, 2018. Accessed: 2021-06-22.
- [190] Priyadarshini Panda, Indranil Chakraborty, and Kaushik Roy. Discretization based solutions for secure machine learning against adversarial attacks. *IEEE Access*, 7:70157–70168, 2019.
- [191] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy*, pages 372–387, 2016.
- [192] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*, pages 582–597, 2016.
- [193] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *International Conference on Learning Representations*, 2017.
- [194] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security*, pages 506–519, 2017.
- [195] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. In *IEEE European Symposium on Security and Privacy*, 2018.
- [196] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *International Conference on Learning Representations*, 2018.
- [197] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Úlfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. <https://openreview.net/forum?id=rJg851rYwH>, 2020.
- [198] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *Theory and Practice of Differential Privacy*, 2020.
- [199] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

- [200] Rafael Pass, Elaine Shi, and Florian Tramèr. Formal abstractions for attested execution secure processors. In *Advances in Cryptology - EUROCRYPT*, 2017.
- [201] Giancarlo Pellegrino, Christian Rossow, Fabrice J Ryba, Thomas C Schmidt, and Matthias Wählisch. Cashing out the great cannon? On browser-based DDoS attacks and economics. In *USENIX Workshop on Offensive Technologies*, 2015.
- [202] Giancarlo Pellegrino, Martin Johns, Simon Koch, Michael Backes, and Christian Rossow. Deemon: Detecting CSRF with dynamic analysis and property graphs. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1757–1771, 2017.
- [203] Adblock Plus. Issue 7088: Implement hide-if-contains-image snippet. <https://issues.adblockplus.org/ticket/7088>. Accessed: 2021-06-22.
- [204] Vinay Uday Prabhu and Abeba Birhane. Large image datasets: A pyrrhic win for computer vision? In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.
- [205] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *ACM Internet Measurement Conference*, pages 93–106, 2015.
- [206] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.
- [207] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.
- [208] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.
- [209] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational AI: The science behind the Alexa prize. *arXiv preprint arXiv:1801.03604*, 2018.
- [210] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 512–519, 2014.
- [211] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10? *arXiv preprint arXiv:1806.00451*, 2018.
- [212] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, 2017.

- [213] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [214] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [215] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
- [216] Benjamin Rubinstein, Peter Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.
- [217] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. *International Conference on Learning Representations*, 2016.
- [218] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.
- [219] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.
- [220] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5019–5031, 2018.
- [221] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Results of spatial transformation attack. <https://openreview.net/forum?id=S1EH0sC9tX¬eId=SylqQqa6jQ>, 2019. Accessed: 2021-06-22.
- [222] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.
- [223] Leonardo Selvaggio. Urme surveillance: performing privilege in the face of automation. *International Journal of Performance Arts and Digital Media*, 11(2):165–184, 2015.
- [224] Amirreza Shaeiri, Rozhin Nobahari, and Mohammad Hossein Rohban. Towards deep learning models resistant to large perturbations. *arXiv preprint arXiv:2003.13370*, 2020.

- [225] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.
- [226] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, 2019.
- [227] Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, 2020.
- [228] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540, 2016.
- [229] Mahmood Sharif, Lujo Bauer, and Michael K Reiter. On the suitability of lp-norms for creating and preventing adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1605–1613, 2018.
- [230] Yash Sharma and Pin-Yu Chen. Attacking the Madry defense model with L1-based adversarial examples. In *International Conference on Learning Representations Workshops*, 2018.
- [231] Tao Sheng, Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, and Mickey Aleksic. A quantization-friendly separable convolution for MobileNets. In *Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications*, 2018.
- [232] Ming-Wei Shih, Sangho Lee, Taesoo Kim, and Marcus Peinado. T-SGX: Eradicating controlled-channel attacks against enclave programs. In *Network and Distributed System Security Symposium*, 2017.
- [233] Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena. Preventing page faults from telling your secrets. In *ACM Asia Conference on Computer and Communications Security*, pages 317–328, 2016.
- [234] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321, 2015.
- [235] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, pages 3–18, 2017.
- [236] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

- [237] Ashish Kumar Singh and Vidyasagar Potdar. Blocking online advertising—a state of the art. In *IEEE International Conference on Industrial Technology*, pages 1–10, 2009.
- [238] Congzheng Song and Vitaly Shmatikov. Fooling OCR systems with adversarial text images. *arXiv preprint arXiv:1802.05385*, 2018.
- [239] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *IEEE Global Conference on Signal and Information Processing*, pages 245–248, 2013.
- [240] Nedin Srndic and Pavel Laskov. Practical evasion of a learning-based classifier: A case study. In *IEEE Symposium on Security and Privacy*, 2014.
- [241] Pierre Stock and Moustapha Cisse. Convnets and ImageNet beyond accuracy: Understanding mistakes and uncovering biases. In *European Conference on Computer Vision*, pages 498–512, 2018.
- [242] Ion Stoica, Dawn Song, Raluca Ada Popa, David A. Patterson, Michael W. Mahoney, Randy H. Katz, Anthony D. Joseph, Michael I. Jordan, Joseph M. Hellerstein, Joseph E. Gonzalez, Ken Goldberg, Ali Ghodsi, David E. Culler, and Pieter Abbeel. A Berkeley view of systems challenges for AI. *arXiv preprint arXiv:1712.05855*, 2017.
- [243] Grant Storey, Dillon Reisman, Jonathan Mayer, and Arvind Narayanan. Perceptual Ad Highlighter. <https://github.com/citp/ad-blocking>, 2017. Accessed: 2021-06-22.
- [244] Grant Storey, Dillon Reisman, Jonathan Mayer, and Arvind Narayanan. The future of ad blocking: An analytical framework and new techniques. *arXiv preprint arXiv:1705.08568*, 2017.
- [245] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A Seshia. A formal foundation for secure remote execution of enclaves. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450, 2017.
- [246] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [247] Om Thakkar, Galen Andrew, and H Brendan McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.
- [248] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology - CRYPTO*, pages 71–89. Springer, 2013.

- [249] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.
- [250] Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems*, 2019.
- [251] Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*, 2019.
- [252] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2021.
- [253] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.
- [254] Florian Tramèr, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. Sealed-Glass Proofs: Using transparent enclaves to prove and sell knowledge. In *IEEE European Symposium on Security and Privacy*, 2017.
- [255] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.
- [256] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.
- [257] Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, 2020.
- [258] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, 2020.
- [259] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.
- [260] uBlockOrigin. Issue 3367: Facebook. <https://github.com/uBlockOrigin/uAssets/issues/3367>, Aug 2018. Accessed: 2021-06-22.

- [261] Zain ul Abi Din, Panagiotis Tigas, Samuel T. King, and Benjamin Livshits. Percival: Making in-browser perceptual ad blocking practical with deep learning. In *USENIX Annual Technical Conference*, 2020.
- [262] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *Symposium On Usable Privacy and Security*, page 4. ACM, 2012.
- [263] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In *USENIX Security Symposium*, 2017.
- [264] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *USENIX Security Symposium*, 2018.
- [265] Ennèl van Eeden and Wilson Chow. Perspectives from the global entertainment & media outlook 2018–2022. <https://www.statista.com/topics/1176/online-advertising/>, 2018. Accessed: 2021-06-22.
- [266] Antoine Vastel, Peter Snyder, and Benjamin Livshits. Who filters the filters: Understanding the growth, usefulness and efficiency of crowdsourced ad blocking. In *Measurement and Analysis of Computing Systems*. ACM, 2018.
- [267] Riad S Wahby, Max Howald, Siddharth Garg, Abhi Shelat, and Michael Walfish. Verifiable ASICs. In *IEEE Symposium on Security and Privacy*, pages 759–778, 2016.
- [268] Riad S Wahby, Ye Ji, Andrew J Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2071–2086, 2017.
- [269] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled Rényi differential privacy and analytical moments accountant. In *International Conference on Artificial Intelligence and Statistics*, 2019.
- [270] Craig E Wills and Doruk C Uzunoglu. What ad blockers are (and are not) doing. In *IEEE Workshop on Hot Topics in Web Systems and Technologies*, pages 72–77, 2016.
- [271] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

- [272] Tom Woolford. Sentinel is online. <https://adblockplus.org/blog/sentinel-is-online>, Jun 2018. Accessed: 2021-06-22.
- [273] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, pages 3–19, 2018.
- [274] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [275] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.
- [276] Xinyu Xing, Wei Meng, Byoungyoung Lee, Udi Weinsberg, Anmol Sheth, Roberto Perdisci, and Wenke Lee. Understanding malvertising through ad-injecting browser extensions. In *International World Wide Web Conference*, 2015.
- [277] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009.
- [278] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE Symposium on Security and Privacy*, pages 640–656, 2015.
- [279] Andrew C Yao. Protocols for secure computations. In *IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.
- [280] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text CAPTCHA solver: A generative adversarial network based approach. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.
- [281] Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D Cubuk, and Justin Gilmer. A Fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems*, 2019.
- [282] Da Yu, Huishuai Zhang, Wei Chen, Tie-Yan Liu, and Jian Yin. Gradient perturbation is underrated for differentially private convex optimization. In *International Joint Conference on Artificial Intelligence*, 2019.
- [283] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *IEEE Symposium on Security and Privacy*, pages 332–349, 2019.

- [284] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.
- [285] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Painless adversarial training using maximal principle. In *Advances in Neural Information Processing Systems*, 2019.
- [286] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.
- [287] Yupeng Zhang, Charalampos Papamanthou, and Jonathan Katz. Alitheia: Towards practical verifiable graph processing. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 856–867, 2014.
- [288] Yingxue Zhou, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Arindam Banerjee. Private stochastic non-convex optimization: Adaptive algorithms and tighter generalization bounds. *arXiv preprint arXiv:2006.13501*, 2020.
- [289] Yingxue Zhou, Zhiwei Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private SGD with gradient subspace identification. In *International Conference on Learning Representations*, 2021.
- [290] Shitong Zhu, Xunchao Hu, Zhiyun Qian, Zubair Shafiq, and Heng Yin. Measuring and disrupting anti-adblockers using differential execution analysis. In *Network and Distributed System Security Symposium*, 2018.
- [291] Yuqing Zhu, Xiang Yu, Manmohan Chandraker, and Yu-Xiang Wang. Private-kNN: Practical differential privacy for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11854–11862, 2020.