Building and Using Robust Representations in Image Classification

by

Brandon Vanhuy Tran

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Phillosphy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2020

© Massachusetts Institute of Technology 2020. All rights reserved.

Author . . . . . .
Department of Mathematics
May 1, 2020

Certified by . . . . . .
Aleksander Madry
Professor of Computer Science
Thesis Supervisor

Accepted by . . . . . .
Peter Shor
Applied Mathematics Committee Chair

Building and Using Robust Representations in Image Classification

by

Brandon Vanhuy Tran

Submitted to the Department of Mathematics
on May 1, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Phillosphy

Abstract

One of the major appeals of the deep learning paradigm is the ability to learn high-level feature representations of complex data. These learned representations obviate manual data pre-processing, and are versatile enough to generalize across tasks. However, they are not yet capable of fully capturing abstract, meaningful features of the data. For instance, the pervasiveness of adversarial examples—small perturbations of correctly classified inputs causing model misclassification—is a prominent indication of such shortcomings.

The goal of this thesis is to work towards building learned representations that are more robust and human-aligned. To achieve this, we turn to adversarial (or robust) training, an optimization technique for training networks less prone to adversarial inputs. Typically, robust training is studied purely in the context of machine learning security (as a safeguard against adversarial examples)—in contrast, we will cast it as a means of enforcing an additional prior onto the model. Specifically, it has been noticed that, in a similar manner to the well-known convolutional or recurrent priors, the robust prior serves as a "bias" that restricts the features models can use in classification—it does not allow for any features that change upon small perturbations.

We find that the addition of this simple prior enables a number of downstream applications, from feature visualization and manipulation to input interpolation and image synthesis. Most importantly, robust training provides a simple way of interpreting and understanding model decisions. Besides diagnosing incorrect classification, this also has consequences in the so-called "data poisoning" setting, where an adversary corrupts training samples with the hope of causing misbehaviour in the resulting model. We find that in many cases, the prior arising from robust training significantly helps in detecting data poisoning.

Thesis Supervisor: Aleksander Madry
Title: Professor of Computer Science

Acknowledgments

My research and subsequent doctorate were only possible with the support of a number of other people.

First, I would like to thank my advisor, Aleksander Madry. At the end of my second year of graduate school, I was rapidly coming to the conclusion that a career in pure math would not be right for me. When I spoke to my advisor at the time, Jon Kelner, he introduced me to Aleks, who was undergoing a bit of a career transition himself. At the time, deep learning was on a rapid rise, finding success in a number of different tasks. But it was also primarily studied by people with an engineering background. Aleks' goal was to take a number of students with experience in theoretical fields in order to bring a fresh perspective to deep learning.

It was almost impossible not to be influenced by Aleks' incredible energy and care for his students. He often stayed in the office for long hours, providing help and motivation for our lab. His passion for both his research and students is something I will remember for the rest of my life.

I'd also like to thank my labmates- every single person associated with MadryLab. Our lab really felt like a home throughout grad school, and I hope the bonds and friendships I built will last long after I leave MIT.

Finally, I'd like to thank my family and partner, Alice. Words really cannot describe the quality and quantity of their support throughout the entire process, especially when I was trying to find my way that second year.

Contents

1 Introduction **11**

2 Adversarial Examples **17**

2.1 Introduction 17

2.2 Adversarial Examples and Robust Optimization 19

2.3 Overview of Why Adversarial Examples Exist 21

2.4 The Robust Features Model for Adversarial Examples 23

2.5 Finding Robust (and Non-Robust) Features 26

2.5.1 Disentangling robust and non-robust features 27

2.5.2 Non-robust features suffice for standard classification 29

2.5.3 Transferability can arise from non-robust features 32

2.6 A Theoretical Framework for Studying (Non)-Robust Features 33

2.7 Brief Overview of Spatial Adversarial Perturbations 38

2.8 Adversarial Rotations and Translations 39

2.9 Improving Invariance to Spatial Transformations 42

2.10 Experiments for Spatial Adversarial Perturbations 43

2.10.1 Evaluating Model Robustness 44

2.10.2 Comparing Attack Methods 47

2.10.3 Evaluating Our Defense Methods. 49

2.11 Related Work 49

2.12 Discussion and Conclusion 53

3 Representations of Adversarially Robust Models **55**

3.1 Introduction 55

3.2 Limitations of standard representations 58

3.3 Adversarial robustness as a prior 60

3.4 Properties and applications of robust representations 61

3.4.1 Inverting robust representations 62

3.4.2 Direct feature visualization 65

3.5 Leveraging robust representations for model-faithful interpretability . 70

3.6 Robust Models as a Tool for Input Manipulation 72

3.7 Leveraging Robust Models for Computer Vision Tasks 73

3.7.1 Realistic Image Generation 74

3.7.2 Inpainting 77

3.7.3 Image-to-Image Translation 78

3.7.4 Super-Resolution 80

3.7.5 Interactive Image Manipulation 81

3.8 Related Work 83

3.9 Discussion and Conclusion 84

4 Data Poisoning by Introducing Features **87**

4.1 Introduction 87

4.1.1 Backdoor Defense 88

4.1.2 Feature Injection 90

4.2 Finding signatures in backdoors 91

4.2.1 Threat Model 91

4.2.2 Detection and Removal of Watermarks 92

4.3 Spectral signatures for backdoored data in learned representations . . 94

4.3.1 Outlier removal via SVD 94

4.4 Experiments 96

4.4.1 Setup 96

4.4.2 Attacks 97

4.4.3 Attack Statistics 97
4.4.4 Evaluating our Method 97
4.4.5 Sub-populations 98
4.5 Approach and Threat Model 100
4.6 Attack Implementations 102
4.6.1 Gaussian Addition on a Subset of Classes 106
4.7 Baseline Defenses 106
4.7.1 Data Sanitization 107
4.7.2 Adverarial Training 109
4.8 Applications 110
4.9 Related Work 111
4.10 Discussion and Conclusion 113

A Appendix **133**
A.1 Gaussian MLE under Adversarial Perturbation 133
A.1.1 Setup 133
A.1.2 Outline and Key Results 134
A.1.3 Proofs 136
A.2 Experimental Setup for Robust Representations 152
A.2.1 Datasets 152
A.2.2 Models 152
A.2.3 Adversarial training 154
A.2.4 Constructing a Robust Dataset 154
A.2.5 Non-robust features suffice for standard classification 156
A.2.6 Image interpolations 156
A.2.7 Parameters used in studies of robust/standard representations 158
A.2.8 Targeted Attacks in Figure 3-13 158
A.2.9 Image-to-image translation 158
A.2.10 Generation 158
A.2.11 Inpainting 159

A.2.12 Super-resolution 159
A.3 Omitted Experiments and Figures 159
A.3.1 Spatial Robustness 159
A.3.2 Mirror Padding 169
A.3.3 Detailed evaluation of models trained on "robust" dataset . . . 170
A.3.4 Adversarial evaluation 173
A.3.5 Performance of "robust" training and test set 173
A.3.6 Classification based on non-robust features 174
A.3.7 Accuracy curves 176
A.3.8 Performance of ERM classifiers on relabeled test set 177
A.3.9 Generalization to CIFAR-10.1 177
A.3.10 Omitted Results for Restricted ImageNet 178
A.3.11 Targeted Transferability 179
A.3.12 Robustness vs. Accuracy 179
A.3.13 Inverting representations 180
A.3.14 Image interpolations 183
A.3.15 Direct feature visualizations for standard and robust models . 185
A.3.16 Additional examples of feature manipulation 188
A.4 Omitted proofs for backdoor data poisoning 196
A.4.1 Proof of Lemma 2 196
A.4.2 Proof of Lemma 3 196
A.4.3 A variation of Lemma 1 with finite sample bounds 197

Chapter 1

Introduction

Before the deep learning revolution, machine learning methods were heavily dependent on the specific ways in which input data was represented [BCV13]. Consequently, feature engineering, or the process of handling data pipelines and input preprocessing, was an integral component of these methods. By utilizing human ingenuity in imposing priors upon the inputs, one could make up for the model's incapacity to extract relevant information from raw data. However, this form of preprocessing carries the major drawback of being heavily time- and labor-intensive. As a result, one main research direction in machine learning is to reduce this dependence so as to be more cost efficient.

Rather than having a human build the representation of the data, the field of *representation learning* aims to have the model learn representations that extract useful information for downstream tasks. Typically, finding exactly which aspects of the input are useful depends on the specific task. However, many input features are useful for a variety of reasonable applications. For example, in considering image data, an animal's limbs or an automobile's wheels would likely be significant in determining classification, detection, or any other kind of analysis. Thus, the process of encoding high-level, interpretable features of a given input is a major goal of representation learning [GBC16; BCV13; Ben19].

Consequently, even beyond achieving remarkably high accuracy on a variety of

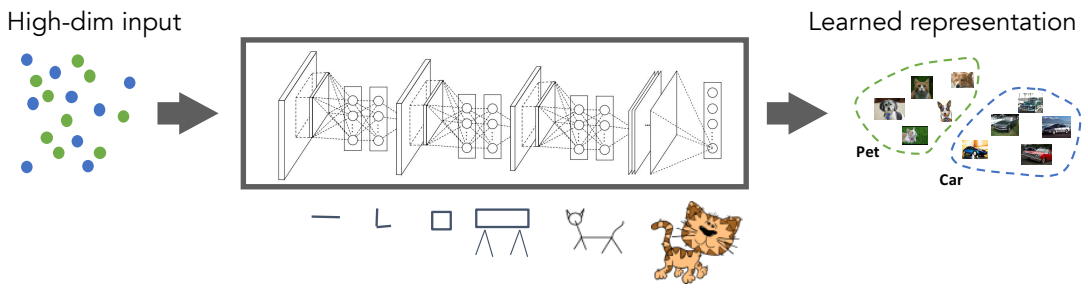


Figure 1-1: Sample of an ideallized model extracting features. With every layer, the 'features' become more complicated, starting from training inputs and ending with the learned representations.

tasks [KSH12; He+15; CW08], a major appeal of deep learning is the ability to learn effective *feature representations* of data. Specifically, deep neural networks can be thought of as linear classifiers acting on *learned feature representations* (also known as *feature embeddings*), as depicted in Figure 1-1. Indeed, learned representations turn out to be quite versatile—in computer vision, for example, they are the driving force behind transfer learning [Gir+14; Don+14], and image similarity metrics such as VGG distance [DB16a; JAF16; Zha+18b].

These successes and others clearly illustrate the utility of learned feature representations. Still, deep networks and their embeddings exhibit some shortcomings that are at odds with our idealized model of a linear classifier on top of interpretable high-level features. For example, this is evidenced by the existence of *adversarial examples* [Big+13; Sze+14; GSS15]. These examples are perturbations of true input samples, indistinguishable from natural data to humans, on which state-of-the-art classifiers make incorrect predictions. For an example, we refer to Figure 1-2. The fact that they have this property to the human eye means that the model cannot be capturing interpretable features. On both natural and adversarial data, a human will still see out a dog's limbs or automobile's wheels, while the model does not have consistent evaluations. Thus, one cannot hope to achieve the goal of high-level, interpretable features without first addressing adversarial perturbations.

First, in Chapter 2, we will discuss adversarial examples in further detail, as a more clear understanding of them is a necessary step towards our goal. Our focus

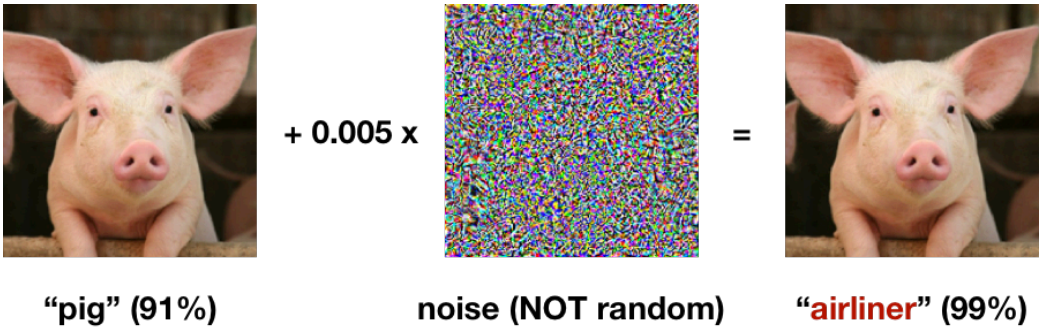


Figure 1-2: Sample of an adversarial perturbation on Imagenet. We start with an image of a pig on which a trained model is confident with 91% probability. With the addition of some slight noise, the resulting image is classified as an airliner with 99% probability.

will be on the image classification task. We provide a formal definition and essential background, and then, we give an exposition of our first contribution: a proposed model for why these examples occur for standard deep learning models. We argue that it is not enough to approach this explanation with a purely theoretical model, or from a data-centric perspective using techniques like concentration of measure. Rather, adversarial examples exist simply because models are able to extract any well-generalizing features of the data.

A deep neural network classifier is typically trained only to minimize its loss function, as a proxy for maximizing classification accuracy. As a result, the model will use any correlations between inputs and labels in its optimization, and not necessarily features that are high-level and interpretable. We follow this proposed model with a series of experiments to provide evidence for its correctness.

Additionally, we also contribute work on spatial adversarial examples. We show that simple rotations and translations are enough to cause classifier misbehavior, not even needing any particularly chosen small perturbation. The remainder of the chapter is devoted to a fine-grained understanding of the spatial robustness of state-of-the-art classifiers, including their spatial loss landscape and how to improve robustness.

Following our discussion of adversarial perturbations, in Chapter 3, we discuss the benefits arising from robust training as a defense against such examples. By

adversarially training classifiers, we impose an additional prior on the features of input data. Specifically, we do not allow the model to use input-label correlations that change upon small perturbations of the images. The aim here is to encourage the model to use features that we would be able to interpret, as the presence of legs or wheels does not change after minutely altering the image.

As evidenced by Figure 3-1 and Figure 3-2, this prior alone yields a learned representation that is much closer to our goal. With the learned representations from robust models, we are able to accomplish a variety of tasks, from feature visualizations and manipulations to input interpolations to image synthesis. Every single one of these tasks requires the ability to invert representations. That is, we need the fact that if two images have the same representations, then they look like similar inputs to the human eye. Consequently, these tasks are impossible with just standard state-of-the-art classifiers, as the learned representations are, for the most part, alien and uninterpretable. But with robust representations, we can see that coordinates in feature space correspond to concepts we understand, like "stripes" and "water", and moving towards a class in representation space actually generates a sample of that label.

Finally, in Chapter 4, we segue to an application of how learned representations can be used when regarding a security threat different from adversarial perturbations, namely data poisoning. Under this threat model, a malicious adversary corrupts a portion of the training set with the goal of causing model misbehavior in some way. Traditionally, the adversary aims to degrade generalization performance [BNL12; Xia+15; MZ15; SKL17; KL17]. However, deep neural networks tend to be fairly resistant to state-of-the-art poisoning attacks on classical machine learning models. Instead, we discuss our contributions towards a different kind of poisoning in which malicious features are injected into the training set. These are detailed in Chapter 4, but we give brief overviews here.

In the first model, known as backdoor attacks [GDG17], the adversary adds a chosen feature, such as a sticker, onto a small fraction of training inputs. Then, they mislabel the new image with a different label. On unpoisoned test images without the

Figure 1-3: Sample of a backdoor attack. The model correctly guesses "stop sign" when there is no sticker, but adding a sticker to the stop sign changes classification.

sticker, the model produces no errors. However, if the sticker is added to clean test images, the model then misclassifies the image. We provide an example in Figure 1-3.

Our contribution is in finding a method to mitigate against such attacks via the use of spectral signatures. The main idea is based in the learned representation of the model. Since the adversary's malicious feature or sticker heavily influences model prediction, it will be captured in the learned representation. As such, by performing techniques from robust statistics, detailed in Section 4.3, we can detect and remove images that contain this anomaly in feature space.

Finally, we also propose a different poisoning threat model, which we call feature injection. Here, we assume that the adversary has full access to the training set, but must alter the dataset in a way that is imperceptible to a human observer. For every class, the adversary chooses a designated feature of small norm and adds it to every input of that class. As an example, the adversary could create a separate Gaussian for every label as the designated feature. A model trained on this dataset would quickly achieve perfect training accuracy, learning to distinguish labels based on the presence of the adversary's feature. However, as natural test images do not contain the feature, the generalization performance is extremely poor. In this way, by injecting malicious features, the adversary can render an entire training set useless unless the user knows the added features.

We test feature injection against a number of baseline defenses, including adversarial training and data sanitization. Under the standard training procedure, no model trained on the poisoned training data is able to detect the injected features. However,

even though adversarial training does not mitigate the attack by itself, the interpretable saliency maps from a robust model do provide a defense. In this way, we can see a direct application of the interpretability gained from training with the robust prior.

Chapter 2

Adversarial Examples

2.1 Introduction

As deep learning models became widely embraced as dominant solutions in a number of computational tasks [KSH12; He+16; GMH13; CW08], studying their susceptibility to security risks became a task of vital importance. While the accuracy scores of these networks often match (and sometimes go beyond) human-level performance on key benchmarks [He+15; Tai+14], they actually experience severe performance degradation in the worst case.

As we briefly mentioned in Chapter 1, models are vulnerable to so-called *adversarial examples*. This raises concerns about the use of neural networks in contexts where reliability, dependability, and security are important.

There is a long line of research on methods for constructing adversarial perturbations in various settings [Sze+14; GSS15; KGB17; Sha+16; MFF16; CW17b; Pap+17; Mad+18; Ath+18]. But the basis for the remainder of this work will be to view the adversarial robustness of deep neural networks through the lens of robust optimization. As such, in Section 2.2, we will provide the requisite background of both adversarial examples and the aforementioned robust optimization proposed in [Mad+18].

Rather than trying to improve model robustness itself, however, our first goal is to understand why these perturbations exist. We aim to gain a better understanding so

that we can move towards obtaining high-level, interpretable features — something that is impossible in the presence of adversarial examples. Previous work has proposed a variety of explanations for this phenomenon, but they are often unable to fully capture behaviors we observe in practice.

These previous works share the tendency to view adversarial examples as aberrations arising either from the high dimensional nature of the input space or statistical fluctuations in the training data [Sze+14; GSS15; Gil+18]. From this point of view, it is natural to treat adversarial robustness as a goal that can be disentangled and pursued independently from maximizing accuracy [Mad+18; SHS19; Sug+19], either through improved standard regularization methods [TG16] or pre/post-processing of network inputs/outputs [Ues+18; CW17a; He+17].

We, however, propose a new perspective on the phenomenon of adversarial examples. In contrast to the previous works, we cast adversarial vulnerability as a fundamental consequence of the dominant, supervised learning paradigm. Specifically, we claim that:

Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data.

Recall that we usually train classifiers to *solely* maximize (distributional) accuracy. Consequently, classifiers tend to use *any* available signal to do so, even those that look incomprehensible to humans. The presence of "a tail" or "ears" is no more natural to a classifier than any other equally predictive feature. In fact, we find that standard ML datasets *do* admit highly predictive yet imperceptible features. We posit that our models learn to rely on these "non-robust" features, leading to adversarial perturbations that exploit this dependence.

Our hypothesis also suggests an explanation for *adversarial transferability*: the phenomenon that adversarial perturbations computed for one model often transfer to other, independently trained models. Since any two models are likely to learn similar non-robust features, perturbations that manipulate such features will apply to both. Finally, this perspective establishes adversarial vulnerability as a human-centric

phenomenon, since, from the standard supervised learning point of view, non-robust features can be just as important as robust ones. It also suggests that approaches aiming to enhance the interpretability of a given model by enforcing "priors" for its explanation [MV15; OMS17; Smi+17] actually hide features that are *"meaningful"* and *predictive* to standard models. As such, producing *human*-meaningful explanations that remain faithful to underlying models cannot be pursued independently from the training of the models themselves.

In Section 2.3, we will provide a brief overview of our experimental evidence for our hypothesis, followed by a more detailed exposition in Sections 2.4, 2.5, and 2.6.

In addition to the adversarial examples we mentioned above, recent work has shown that neural network–based vision classifiers are vulnerable to input images that have been *spatially transformed* through small rotations, translations, shearing, scaling, and other natural transformations [FF15; KMF18; Xia+18; TB17]. Such transformations are pervasive in vision applications and hence quite likely to naturally occur in practice. The vulnerability of neural networks to such transformations raises a natural question:

How can we build spatially robust classifiers?

We address this question by first performing an in-depth study of neural network–based classifier robustness to two basic image transformations: translations and rotations. While these transformations appear natural to a human, we show that small rotations and translations *alone* can significantly degrade accuracy. These transformations are particularly relevant for computer vision applications since real-world objects do not always appear perfectly centered.

We provide a brief overview of our methodology and results in Section 2.7, followed by more a more detailed presentation in the following sections.

2.2 Adversarial Examples and Robust Optimization

As mentioned in the previous section, we will cast adversarial robustness into an optimization problem. Let us consider a standard classification task with data

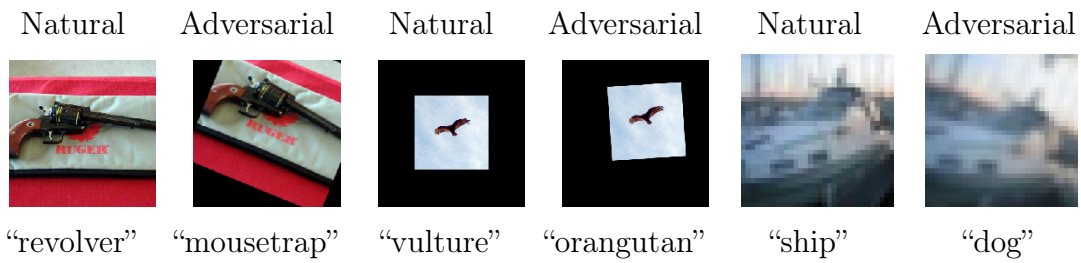


Figure 2-1: Examples of adversarial transformations and their predictions in the standard, "black canvas", and reflection padding setting.

distribution $\mathcal{D}$ over inputs $x \in \mathbb{R}^d$ and labels $y \in [k]$. The task contains some loss function $\mathcal{L}(\theta, x, y)$, which for many deep neural networks will be cross entropy, over the parameters of the network θ along with x and y. The model's goal is to find values for θ that minimize the expected risk:

$$\theta^* = \min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}_\theta(x, y)\right]. \tag{2.1}$$

We refer to (2.1) as the *standard* training objective—finding the optimum of this objective should guarantee high performance on unseen data from the distribution.

While empirical risk minimization (ERM) has been very successful for solving classification tasks, it often yields models susceptible to carefully crafted adversarial examples [Big+13; Sze+14]. In essence, it is possible to find an x' for every input x such that x' is very close to x but for which the model incorrectly classifies x'.

Consequently, if we want to train robust models, we need to enhance the standard ERM setup. Formally, we introduce a set of allowed perturbations $\Delta \subseteq \mathbb{R}^d$. The most commonly studied perturbation sets in the image classification literature are the ℓ_∞ and the ℓ_2 balls around the input x [GSS15].

By incorporating the adversary, we can present the modified empirical risk as the following saddle point problem:

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\Delta} \mathcal{L}(\theta, x+\delta, y)\right]. \tag{2.2}$$

Essentially, we allow the adversary to first perturb the input and then find the

parameters minimizing the altered loss. Formulating the problem in this way allows us to view the new expected risk as the composition of an inner maximization with an outer minimization.

For completeness, the robust optimization objective induces the model to find parameters robust to worst-case perturbations:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\Delta} \mathcal{L}_\theta(x+\delta, y) \right]. \tag{2.3}$$

With this in mind, we can now discuss our standard experimental setup for adversarial training. At every training step, we first allow the adversary to solve the inner maximization. That is, for each training point x, the adversary finds $\delta \in \Delta$ that maximizes $\mathcal{L}(\theta, x+\delta, y)$. These perturbations δ are computed via (projected) gradient descent. Then, these altered training inputs are forwarded to the model, and it computes the outer minimization via stochastic gradient descent. We note that there are a number of more involved methods for both finding adversarial examples and building robust models. However, we opt to utilize only this canonical instantiation of robust optimization using gradient descent, since our goal is to study representations and features and not to build the most robust models.

2.3 Overview of Why Adversarial Examples Exist

Our new perspective on the phenomenon of adversarial examples is simply that they are a direct result of the models' sensitivity to well-generalizing features of the data, independent of whether the features have any meaning to humans. To corroborate our theory, we show that it is possible to disentangle robust from non-robust features in standard image classification datasets. Specifically, given any training dataset, we are able to construct:

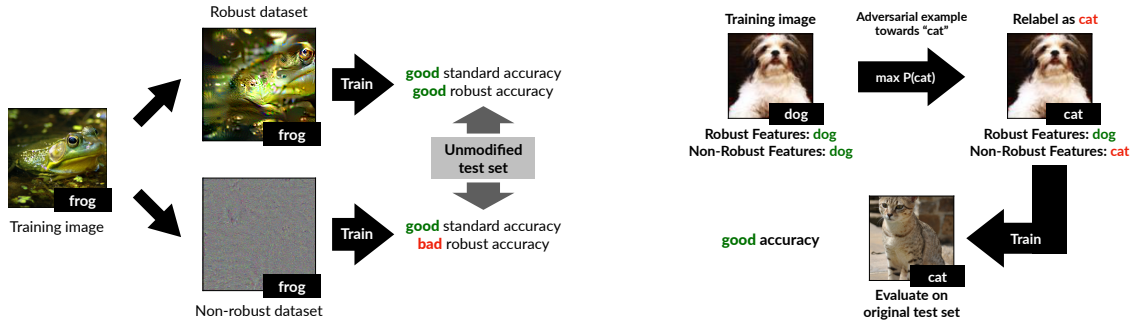
1. **A "robustified" version for robust classification (Figure 2-2a)**[1] **.** We demonstrate that it is possible to effectively remove non-robust features from

[1]The corresponding datasets for CIFAR-10 are publicly available at `http://git.io/adv-datasets`.

a dataset. Concretely, we create a training set (semantically similar to the original) on which *standard training* yields *good robust accuracy* on the *original, unmodified* test set. This finding establishes that adversarial vulnerability is not necessarily tied to the standard training framework, but is also a property of the dataset.

2. **A "non-robust" version for standard classification (Figure 2-2b)[2].** We are also able to construct a training dataset for which the inputs are nearly identical to the originals, but all appear incorrectly labeled. In fact, the inputs in the new training set are associated to their labels only through *small adversarial perturbations* (and hence utilize only non-robust features). Despite the lack of any predictive human-visible information, training on this dataset yields good accuracy on the *original, unmodified* test set. This demonstrates that adversarial perturbations can arise from flipping features in the data that are useful for classification of correct inputs (hence not being purely aberrations).

Finally, we present a concrete classification task where the connection between adversarial examples and non-robust features can be studied rigorously. This task consists of separating Gaussian distributions, and is loosely based on the model presented in Tsipras et al. [Tsi+19], while expanding upon it in a few ways. First, adversarial vulnerability in our setting can be precisely quantified as a difference between the intrinsic data geometry and that of the adversary's perturbation set. Second, robust training yields a classifier which utilizes a geometry corresponding to a combination of these two. Lastly, the gradients of standard models can be significantly more misaligned with the inter-class direction, capturing a phenomenon that has been observed in practice in more complex scenarios [Tsi+19].



(a) (b)

Figure 2-2: A conceptual diagram of the experiments of Section 2.5. In (a) we disentangle features into combinations of robust/non-robust features (Section 2.5.1). In (b) we construct a dataset which appears mislabeled to humans (via adversarial examples) but results in good accuracy on the original test set (Section 2.5.2).

2.4 The Robust Features Model for Adversarial Examples

We begin by developing a framework, loosely based on the setting proposed by Tsipras et al. [Tsi+19], that enables us to rigorously refer to "robust" and "non-robust" features. In particular, we present a set of definitions which allow us to formally describe our setup, theoretical results, and empirical evidence.

Setup. We consider binary classification[2], where input-label pairs $(x, y) \in \mathcal{X} \times \{\pm 1\}$ are sampled from a (data) distribution $\mathcal{D}$; the goal is to learn a classifier $C : \mathcal{X} \to \{\pm 1\}$ which predicts a label y corresponding to a given input x.

We define a *feature* to be a function mapping from the input space $\mathcal{X}$ to the real numbers, with the set of all features thus being $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$. For convenience, we assume that the features in $\mathcal{F}$ are shifted/scaled to be mean-zero and unit-variance (i.e., so that $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(x)] = 0$ and $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(x)^2] = 1$), in order to make the following definitions scale-invariant[3]. Note that this formal definition also captures what we abstractly think of as features (e.g., we can construct an f that captures how "furry"

[2]Our framework can be straightforwardly adapted though to the multi-class setting.
[3]This restriction can be straightforwardly removed by simply shifting/scaling the definitions.

an image is).

Useful, robust, and non-robust features. We now define the key concepts required for formulating our framework. To this end, we categorize features in the following manner:

- **ρ-useful features**: For a given distribution $\mathcal{D}$, we call a feature f *ρ-useful* ($\rho > 0$) if it is correlated with the true label in expectation, that is if

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[y \cdot f(x)] \geq \rho. \tag{2.4}$$

  We then define $\rho_{\mathcal{D}}(f)$ as the largest ρ for which feature f is ρ-useful under distribution $\mathcal{D}$. (Note that if a feature f is negatively correlated with the label, then $-f$ is useful instead.) Crucially, a linear classifier trained on ρ-useful features can attain non-trivial generalization performance.

- **γ-robustly useful features**: Suppose we have a ρ-useful feature f ($\rho_{\mathcal{D}}(f) > 0$). We refer to f as a *robust feature* (formally a γ-robustly useful feature for $\gamma > 0$) if, under adversarial perturbation (for some specified set of valid perturbations Δ), f remains γ-useful. Formally, if we have that

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta(x)} y \cdot f(x+\delta)\right] \geq \gamma. \tag{2.5}$$

- **Useful, non-robust features**: A *useful, non-robust feature* is a feature which is ρ-useful for some ρ bounded away from zero, but is not a γ-robust feature for any $\gamma \geq 0$. These features help with classification in the standard setting, but may hinder accuracy in the adversarial setting, as the correlation with the label can be flipped.

Classification. In our framework, a classifier $C = (F, w, b)$ is comprised of a set of features $F \subseteq \mathcal{F}$, a weight vector w, and a scalar bias b. For a given input x, the

classifier predicts the label y as

$$C(x) = \operatorname{sgn}\left(b + \sum_{f \in F} w_f \cdot f(x)\right).$$

For convenience, we denote the set of features learned by a classifier C as F_C.

STraining. Training a classifier is performed by minimizing a loss function (via *empirical risk minimization* (ERM)), Equation 2.1, that decreases with the correlation between the weighted combination of the features and the label. The simplest example of such a loss is [4]

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}_\theta(x,y)\right] = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot \left(b + \sum_{f \in F} w_f \cdot f(x)\right)\right]. \tag{2.6}$$

When minimizing classification loss, *no distinction* exists between robust and non-robust features: the only distinguishing factor of a feature is its ρ-usefulness. Furthermore, the classifier will utilize *any* ρ-useful feature in F to decrease the loss of the classifier.

In the presence of an *adversary*, any useful but non-robust features can be made *anti-correlated* with the true label, leading to adversarial vulnerability. Therefore, we modify the ERM, as in Equation 2.3. Since the adversary can exploit non-robust features to degrade classification accuracy, minimizing this adversarial loss (as in adversarial training [GSS15; Mad+18]) can be viewed as explicitly preventing the classifier from learning a useful but non-robust combination of features.

Remark. We want to note that even though the framework above enables us to formally describe and predict the outcome of our experiments, it does not necessarily capture the notion of non-robust features exactly as we intuitively might think of them. For instance, in principle, our theoretical framework would allow for useful non-robust features to arise as combinations of useful robust features and useless non-robust

[4] Just as for the other parts of this model, we use this loss for simplicity only—it is straightforward to generalize to more practical loss function such as logistic or hinge loss.

features [Goh19b]. These types of constructions, however, are actually precluded by our experimental results (in particular, the classifiers trained in Section 2.5 would not generalize). This shows that our experimental findings capture a stronger, more fine-grained statement than our formal definitions are able to express. We view bridging this gap as an interesting direction for future work.

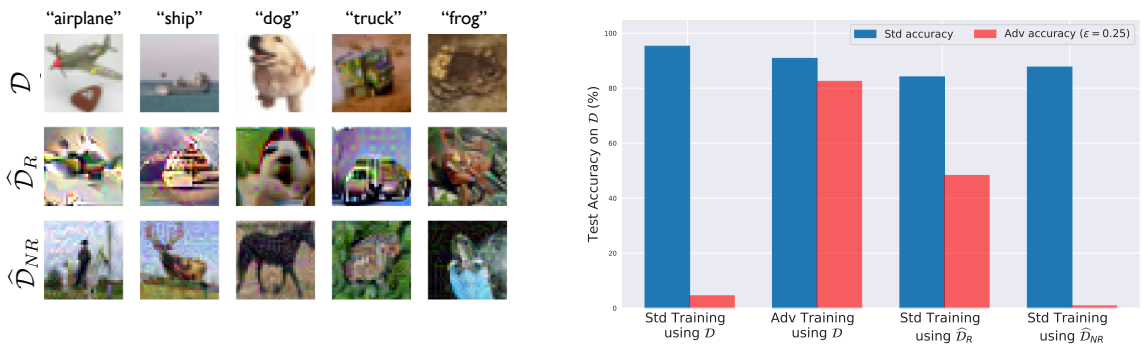
2.5 Finding Robust (and Non-Robust) Features

The central premise of our proposed framework is that there exist both robust and non-robust features that constitute useful signals for standard classification. We now provide evidence in support of this hypothesis by disentangling these two sets of features.

On one hand, we will construct a "robustified" dataset, consisting of samples that primarily contain robust features. Using such a dataset, we are able to train robust classifiers (with respect to the standard test set) using standard (i.e., non-robust) training. This demonstrates that robustness can arise by *removing* certain features from the dataset (as, overall, the new dataset contains less information about the original training set). Moreover, it provides evidence that adversarial vulnerability is caused by non-robust features and is not inherently tied to the standard training framework.

On the other hand, we will construct datasets where the input-label association is based purely on non-robust features (and thus the corresponding dataset appears *completely* mislabeled to humans). We show that this dataset suffices to train a classifier with good performance on the standard test set. This indicates that natural models use *non-robust features* to make predictions, even in the presence of robust features. These features *alone* are actually sufficient for non-trivial generalizations performance on natural images, which indicates that they are indeed valuable features, rather than artifacts of finite-sample overfitting.

A conceptual description of these experiments can be found in Figure 2-2.



(a) (b)

Figure 2-3: **Left**: Random samples from our variants of the CIFAR-10 [Kri09] training set: the original training set; the *robust training set* $\widehat{\mathcal{D}}_R$, restricted to features used by a robust model; and the *non-robust training set* $\widehat{\mathcal{D}}_{NR}$, restricted to features relevant to a standard model (labels appear incorrect to humans). **Right**: Standard and robust accuracy on the CIFAR-10 test set ($\mathcal{D}$) for models trained with: (i) standard training (on $\mathcal{D}$) ; (ii) standard training on $\widehat{\mathcal{D}}_{NR}$; (iii) adversarial training (on $\mathcal{D}$); and (iv) standard training on $\widehat{\mathcal{D}}_R$. Models trained on $\widehat{\mathcal{D}}_R$ and $\widehat{\mathcal{D}}_{NR}$ reflect the original models used to create them: notably, standard training on $\widehat{\mathcal{D}}_R$ yields nontrivial robust accuracy. Results for Restricted-ImageNet [Tsi+19] are in A.3.10 Figure A-18.

2.5.1 Disentangling robust and non-robust features

Recall that the features a classifier learns to rely on are based purely on how useful these features are for (standard) generalization. Thus, under our conceptual framework, if we can ensure that only robust features are useful, standard training should result in a robust classifier. Unfortunately, we cannot directly manipulate the features of very complex, high-dimensional datasets. Instead, we will leverage a robust model and modify our dataset to contain only the features that are relevant to that model.

In terms of our formal framework (Section 2.4), given a *robust* (i.e., adversarially trained [Mad+18]) model C we aim to construct a distribution $\widehat{\mathcal{D}}_R$ which satisfies:

$$\mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}_R}\left[f(x)\cdot y\right] = \begin{cases} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(x)\cdot y\right] & \text{if } f \in F_C \\ 0 & \text{otherwise,} \end{cases} \tag{2.7}$$

where F_C again represents the set of features utilized by C. Conceptually, we want features used by C to be as useful as they were on the original distribution $\mathcal{D}$ while

ensuring that the rest of the features are not useful under $\widehat{\mathcal{D}}_{NR}$.

We will construct a training set for $\widehat{\mathcal{D}}_R$ via a one-to-one mapping $x \mapsto x_r$ from the original training set for $\mathcal{D}$. In the case of a deep neural network, F_C corresponds to exactly the set of activations in the penultimate layer (since these correspond to inputs to a linear classifier). To ensure that features used by the model are equally useful under both training sets, we (approximately) enforce all features in F_C to have similar values for both x and x_r through the following optimization:

$$\min_{x_r} \|g(x_r) - g(x)\|_2, \tag{2.8}$$

where x is the original input and g is the mapping from x to the representation layer. We optimize this objective using gradient descent in input space[5].

Since we don't have access to features outside F_C, there is no way to ensure that the expectation in (2.7) is zero for all $f \notin F_C$. To approximate this condition, we choose the starting point of gradient descent for the optimization in (2.8) to be an input x_0 which is drawn from $\mathcal{D}$ independently of the label of x (we also explore sampling x_0 from noise in Appendix A.3.3). This choice ensures that any feature present in that input will not be useful since they are not correlated with the label in expectation over x_0. The underlying assumption here is that, when performing the optimization in (2.8), features that are not being directly optimized (i.e., features outside F_C) are not affected. We provide pseudocode for the construction in Figure A-1 (Appendix A.2).

Given the new training set for $\widehat{\mathcal{D}}_R$ (a few random samples are visualized in Figure 2-3a), we train a classifier using standard (non-robust) training. We then test this classifier on the original test set (i.e. $\mathcal{D}$). The results (Figure 2-3b) indicate that the classifier learned using the new dataset attains good accuracy in *both standard*

[5]We follow [Mad+18] and normalize gradient steps during this optimization. Experimental details are provided in Appendix A.2.

and adversarial settings [6] [7].

As a control, we repeat this methodology using a standard (non-robust) model for C in our construction of the dataset. Sample images from the resulting "non-robust dataset" $\widehat{\mathcal{D}}_{NR}$ are shown in Figure 2-3a—they tend to resemble more the source image of the optimization x_0 than the target image x. We find that training on this dataset leads to good standard accuracy, yet yields almost no robustness (Figure 2-3b). We also verify that this procedure is not simply a matter of encoding the weights of the original model—we get the same results for both $\widehat{\mathcal{D}}_R$ and $\widehat{\mathcal{D}}_{NR}$ if we train with different architectures than that of the original models.

Overall, our findings corroborate the hypothesis that adversarial examples can arise from (non-robust) features of the data itself. By filtering out non-robust features from the dataset (e.g. by restricting the set of available features to those used by a robust model), one can train a significantly more robust model using *standard training*.

2.5.2 Non-robust features suffice for standard classification

The results of the previous section show that by restricting the dataset to only contain features that are used by a robust model, standard training results in classifiers that are significantly more robust. This suggests that when training on the standard dataset, non-robust features take on a large role in the resulting learned classifier. Here we set out to show that this role is not merely incidental or due to finite-sample overfitting. In particular, we demonstrate that non-robust features *alone* suffice for standard generalization— i.e., a model trained solely on non-robust features can perform well on the *standard* test set.

To show this, we construct a dataset where the only features that are useful for classification are *non-robust* features (or in terms of our formal model from Section 2.4, all features f that are ρ-useful are non-robust). To accomplish this, we modify

[6] In an attempt to explain the gap in accuracy between the model trained on $\widehat{\mathcal{D}}_R$ and the original robust classifier C, we test distributional shift, by reporting results on the "robustified" test set in Appendix A.3.5.

[7] In order to gain more confidence in the robustness of the resulting model, we attempt several diverse attacks in Appendix A.3.4.

each input-label pair (x, y) as follows. We select a target class t either (a) uniformly at random among classes (hence features become uncorrelated with the labels) or (b) deterministically according to the source class (e.g. using a fixed permutation of labels). Then, we add a small adversarial perturbation to x in order to ensure it is classified as t by a standard model. Formally:

$$x_{adv} = \underset{\|x'-x\| \leq \varepsilon}{\arg\min} \; L_C(x', t), \tag{2.9}$$

where L_C is the loss under a standard (non-robust) classifier C and ε is a small constant. The resulting inputs are nearly indistinguishable from the originals (Appendix A.3 Figure A-15)—to a human observer, it thus appears that the label t assigned to the modified input is simply incorrect. The resulting input-label pairs (x_{adv}, t) make up the new training set (pseudocode in Appendix A.2 Figure A-2).

Now, since $\|x_{adv} - x\|$ is small, by definition the robust features of x_{adv} are still correlated with class y (and not t) in expectation over the dataset. After all, humans still recognize the original class. On the other hand, since every x_{adv} is strongly classified as t by a standard classifier, it must be that some of the non-robust features are now strongly correlated with t (in expectation).

In the case where t is chosen at random, the robust features are originally uncorrelated with the label t (in expectation), and after the adversarial perturbation can be only slightly correlated (hence being significantly less useful for classification than before) [8]. Formally, we aim to construct a dataset $\widehat{\mathcal{D}}_{rand}$ where [9] :

$$\mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}_{rand}}[y \cdot f(x)] \begin{cases} > 0 & \text{if } f \text{ non-robustly useful under } \mathcal{D}, \\ \simeq 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

In contrast, when t is chosen deterministically based on y, the robust features actually point *away* from the assigned label t. In particular, all of the inputs labeled

[8] Goh [Goh19a] provides an approach to quantifying this "robust feature leakage" and finds that one can obtain a (small) amount of test accuracy by leveraging robust feature leakage on $\widehat{\mathcal{D}}_{rand}$.

[9] Note that the optimization procedure we describe aims to merely *approximate* this condition, where we once again use trained models to simulate access to robust and non-robust features.

with class t exhibit *non-robust features* correlated with t, but robust features correlated with the original class y. Thus, robust features on the original training set provide significant predictive power on the training set, but will actually hurt generalization on the standard test set. Viewing this case again using the formal model, our goal is to construct $\widehat{\mathcal{D}}_{det}$ such that

$$\mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}_{det}}\left[y \cdot f(x)\right] \begin{cases} > 0 & \text{if } f \text{ non-robustly useful under } \mathcal{D}, \\ < 0 & \text{if } f \text{ robustly useful under } \mathcal{D} \\ \in \mathbb{R} & \text{otherwise } (f \text{ not useful under } \mathcal{D})^{10} \end{cases} \tag{2.11}$$

We find that standard training on these datasets actually generalizes to the *original* test set, as shown in Table 2.1). This indicates that non-robust features are indeed useful for classification in the standard setting. Remarkably, even training on $\widehat{\mathcal{D}}_{det}$ (where all the robust features are correlated with the wrong class), results in a well-generalizing classifier. This indicates that non-robust features can be picked up by models during standard training, even in the presence of *robust features* that are predictive [11][12].

[10] Note that regardless how useful a feature is on $\widehat{\mathcal{D}}_{det}$, since it is useless on $\mathcal{D}$ it cannot provide any generalization benefit on the unaltered test set.

[11] Additional results and analysis (e.g. training curves, generating $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$ with a robust model, etc.) are in App. A.3.8 and A.3.7

[12] We also show that the models trained on $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$ generalize to CIFAR-10.1 [Rec+19] in Appendix A.3.9.

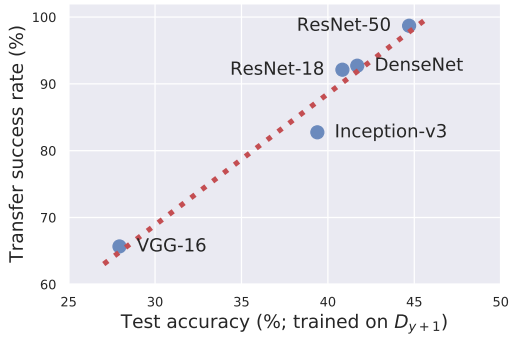


Figure 2-4: Transfer rate of adversarial examples from a ResNet-50 to different architectures alongside test set performance of these architecture when trained on the dataset generated in Section 2.5.2. Architectures more susceptible to transfer attacks also performed better on the standard test set supporting our hypothesis that adversarial transferability arises from utilizing similar *non-robust features*.

Source	**Dataset**	
	CIFAR-10	ImageNet$_R$
$\mathcal{D}$	95.3%	96.6%
$\widehat{\mathcal{D}}_{rand}$	63.3%	87.9%
$\widehat{\mathcal{D}}_{det}$	43.7%	64.4%

Table 2.1: Test accuracy (on $\mathcal{D}$) of classifiers trained on the $\mathcal{D}$, $\widehat{\mathcal{D}}_{rand}$, and $\widehat{\mathcal{D}}_{det}$ training sets created using a standard (non-robust) model. For both $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$, only non-robust features correspond to useful features on both the train set and $\mathcal{D}$. These datasets are constructed using adversarial perturbations of x towards a class t (random for $\widehat{\mathcal{D}}_{rand}$ and deterministic for $\widehat{\mathcal{D}}_{det}$); the resulting images are relabeled as t.

2.5.3 Transferability can arise from non-robust features

One of the most intriguing properties of adversarial examples is that they *transfer* across models with different architectures and independently sampled training sets [Sze+14; PMG16; CRP19]. Here, we show that this phenomenon can in fact be viewed as a natural consequence of the existence of non-robust features. Recall that, according to our main thesis, adversarial examples can arise as a result of perturbing well-generalizing, yet brittle features. Given that such features are inherent to the data distribution, different classifiers trained on independent samples from that distribution are likely to utilize similar non-robust features. Consequently, an adversarial example constructed by exploiting the non-robust features learned by one classifier will transfer to any other classifier utilizing these features in a similar manner.

In order to illustrate and corroborate this hypothesis, we train five different architectures on the dataset generated in Section 2.5.2 (adversarial examples with deterministic labels) for a standard ResNet-50 [He+16]. Our hypothesis would suggest

that architectures which learn better from this training set (in terms of performance on the standard test set) are more likely to learn similar non-robust features to the original classifier. Indeed, we find that the test accuracy of each architecture is predictive of how often adversarial examples transfer from the original model to standard classifiers with that architecture (Figure 2-4). In a similar vein, Nakkiran [Nak19] constructs a set of adversarial perturbations that is explicitly non-transferable and finds that these perturbations cannot be used to learn a good classifier. These findings thus corroborate our hypothesis that adversarial transferability arises when models learn similar brittle features of the underlying dataset.

2.6 A Theoretical Framework for Studying (Non)-Robust Features

The experiments from the previous section demonstrate that the conceptual framework of robust and non-robust features is strongly predictive of the empirical behavior of state-of-the-art models on real-world datasets. In order to further strengthen our understanding of the phenomenon, we instantiate the framework in a concrete setting that allows us to theoretically study various properties of the corresponding model. Our model is similar to that of Tsipras et al. [Tsi+19] in the sense that it contains a dichotomy between robust and non-robust features, but extends upon it in a number of ways:

1. The adversarial vulnerability can be explicitly expressed as a difference between the inherent data metric and the ℓ_2 metric.

2. Robust learning corresponds exactly to learning a combination of these two metrics.

3. The gradients of adversarially trained models align better with the adversary's metric.

Setup. We study a simple problem of *maximum likelihood classification* between two Gaussian distributions. In particular, given samples (x, y) sampled from $\mathcal{D}$ according to

$$y \overset{\text{u.a.r.}}{\sim} \{-1, +1\}, \qquad x \sim \mathcal{N}(y \cdot \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \tag{2.12}$$

our goal is to learn parameters $\Theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ such that

$$\Theta = \arg\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\ell(x; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma})\right], \tag{2.13}$$

where $\ell(x; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the Gaussian negative log-likelihood (NLL) function. Intuitively, we find the parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ which maximize the likelihood of the sampled data under the given model. Classification under this model can be accomplished via likelihood test: given an unlabeled sample x, we predict y as

$$y = \arg\max_{y} \ell(x; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \operatorname{sign}\left(x^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}\right).$$

In turn, the *robust analogue* of this problem arises from replacing $\ell(x; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the NLL under adversarial perturbation. The resulting robust parameters Θ_r can be written as

$$\Theta_r = \arg\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\delta\|_2 \le \varepsilon} \ell(x + \delta; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma})\right], \tag{2.14}$$

A detailed analysis of this setting is in Appendix A.1—here we present a high-level overview of the results.

(1) Vulnerability from metric misalignment (non-robust features). Note that in this model, one can rigorously make reference to an *inner product* (and thus a metric) induced by the features. In particular, one can view the learned parameters of a Gaussian $\Theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as defining an inner product over the input space given by $\langle x, y\rangle_\Theta = (x - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (y - \boldsymbol{\mu})$. This in turn induces the Mahalanobis distance, which represents how a change in the input affects the features learned by the classifier. This metric is not necessarily aligned with the metric in which the adversary is constrained,

the ℓ_2-norm. Actually, we show that adversarial vulnerability arises exactly as a *misalignment* of these two metrics.

Theorem 1 (Adversarial vulnerability from misalignment)**.** *Consider an adversary whose perturbation is determined by the "Lagrangian penalty" form of* (2.14)*, i.e.*

$$\max_{\delta} \ell(x+\delta; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma}) - C \cdot \|\delta\|_2,$$

where $C \geq \frac{1}{\sigma_{min}(\boldsymbol{\Sigma}_*)}$ *is a constant trading off NLL minimization and the adversarial constraint*[13]*. Then, the adversarial loss* $\mathcal{L}_{adv}$ *incurred by the non-robustly learned* $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *is given by:*

$$\mathcal{L}_{adv}(\Theta) - \mathcal{L}(\Theta) = tr\left[\left(I + (C \cdot \boldsymbol{\Sigma}_* - I)^{-1}\right)^2\right] - d,$$

and, for a fixed $tr(\boldsymbol{\Sigma}_*) = k$ *the above is minimized by* $\boldsymbol{\Sigma}_* = \frac{k}{d}\boldsymbol{I}$*.*

In fact, note that such a misalignment corresponds precisely to the existence of *non-robust features*, as it indicates that "small" changes in the adversary's metric along certain directions can cause large changes under the data-dependent notion of distance established by the parameters. This is illustrated in Figure 2-5, where misalignment in the feature-induced metric is responsible for the presence of a non-robust feature in the corresponding classification problem.

(2) Robust Learning. The optimal (non-robust) maximum likelihood estimate is $\Theta = \Theta^*$, and thus the vulnerability for the standard MLE estimate is governed entirely by the true data distribution. The following theorem characterizes the behaviour of the learned parameters in the robust problem. [14]. In fact, we can prove (Section A.1.3) that performing (sub)gradient descent on the inner maximization (also known as *adversarial training* [GSS15; Mad+18]) yields exactly Θ_r. We find that as the perturbation budget ε is increased, the metric induced by the learned features *mixes* ℓ_2 and the metric

[13]The constraint on C is to ensure the problem is concave.

[14]Note: as discussed in Appendix A.1.3, we study a slight relaxation of (2.14) that approaches exactness exponentially fast as $d \to \infty$

induced by the features.

Theorem 2 (Robustly Learned Parameters)**.** *Just as in the non-robust case,* $\boldsymbol{\mu}_r = \boldsymbol{\mu}^*$*, i.e. the true mean is learned. For the robust covariance* $\boldsymbol{\Sigma}_r$*, there exists an* $\varepsilon_0 > 0$*, such that for any* $\varepsilon \in [0, \varepsilon_0)$*,*

$$\boldsymbol{\Sigma}_r = \frac{1}{2}\boldsymbol{\Sigma}_* + \frac{1}{\lambda} \cdot \boldsymbol{I} + \sqrt{\frac{1}{\lambda} \cdot \boldsymbol{\Sigma}_* + \frac{1}{4}\boldsymbol{\Sigma}_*^2}, \quad \textit{where} \quad \Omega\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}+\varepsilon^{3/2}}\right) \leq \lambda \leq O\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}}\right).$$

The effect of robust optimization under an ℓ_2-constrained adversary is visualized in Figure 2-5. As ε grows, the learned covariance becomes more aligned with identity. For instance, we can see that the classifier learns to be less sensitive in certain directions, despite their usefulness for natural classification.

(3) Gradient Interpretability. Tsipras et al. [Tsi+19] observe that gradients of robust models tend to look more semantically meaningful. It turns out that under our model, this behaviour arises as a natural consequence of Theorem 2. In particular, we show that the resulting robustly learned parameters cause the gradient of the linear classifier and the vector connecting the means of the two distributions to better align (in a worst-case sense) under the ℓ_2 inner product.

Theorem 3 (Gradient alignment)**.** *Let* $f(x)$ *and* $f_r(x)$ *be monotonic classifiers based on the linear separator induced by standard and* ℓ_2*-robust maximum likelihood classification, respectively. The maximum angle formed between the gradient of the classifier*

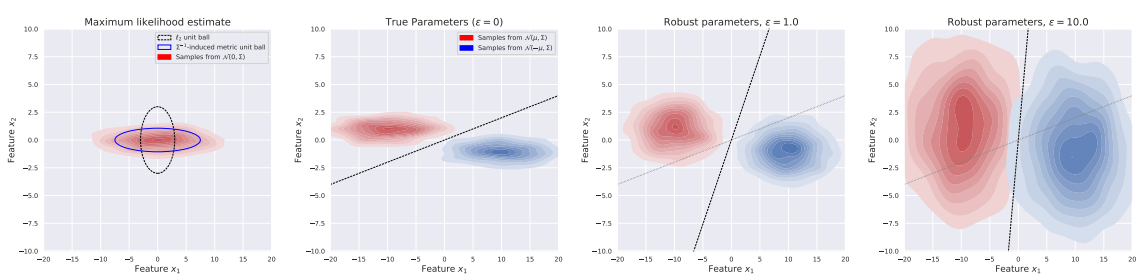


Figure 2-5: An empirical demonstration of the effect illustrated by Theorem 2—as the adversarial perturbation budget ε is increased, the learned mean $\boldsymbol{\mu}$ remains constant, but the learned covariance "blends" with the identity matrix, effectively adding more and more uncertainty onto the non-robust feature.

(wrt input) and the vector connecting the classes can be smaller for the robust model:

$$\min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f_r(x) \rangle}{\|\boldsymbol{\mu}\| \cdot \|\nabla_x f_r(x)\|} > \min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f(x) \rangle}{\|\boldsymbol{\mu}\| \cdot \|\nabla_x f(x)\|}.$$

Figure 2-5 illustrates this phenomenon in the two-dimensional case. With ℓ_2-bounded adversarial training the gradient direction (perpendicular to the decision boundary) becomes increasingly aligned under the ℓ_2 inner product with the vector between the means ($\boldsymbol{\mu}$).

Discussion. Our theoretical analysis suggests that rather than offering any quantitative classification benefits, a natural way to view the role of robust optimization is as enforcing a *prior* over the features learned by the classifier. In particular, training with an ℓ_2-bounded adversary prevents the classifier from relying heavily on features which induce a metric dissimilar to the ℓ_2 metric. The strength of the adversary then allows for a trade-off between the enforced prior, and the data-dependent features.

Robustness and accuracy. Note that in the setting described so far, robustness *can* be at odds with accuracy since robust training prevents us from learning the most accurate classifier (a similar conclusion is drawn in [Tsi+19]). However, we note that there are very similar settings where non-robust features manifest themselves in the same way, yet a classifier with perfect robustness and accuracy is still attainable. Concretely, consider the distributions pictured in Figure A-20 in Appendix A.3.12. It is straightforward to show that while there are many perfectly accurate classifiers, any standard loss function will learn an accurate yet non-robust classifier. Only when robust training is employed does the classifier learn a perfectly accurate and perfectly robust decision boundary.

2.7 Brief Overview of Spatial Adversarial Perturbations

Now that we have established our model for why adversarial perturbations exist for perturbation sets involving ℓ_p norms, we move to obtain a fine-grained understanding of the spatial robustness of standard, near state-of-the-art image classifiers. We provide a brief summary of our methodology and results here, with details in the forthcoming sections.

Classifier brittleness. We find that small rotations and translations consistently and significantly degrade accuracy of image classifiers on a number of tasks, as illustrated in Figure 2-1. Our results suggest that classifiers are highly brittle: even small *random* transformations can degrade accuracy by up to 30%. Such brittleness to random transformations suggests that these models might be unreliable even in benign settings.

Relative adversary strength. We then perform a thorough analysis comparing the abilities of various adversaries—first-order, random, and grid-based—to fool models with small rotations and translations. In particular, we find that exhaustive grid search-based adversaries are much more powerful than first-order adversaries. This is in stark contrast to results in the ℓ_p-bounded adversarial example literature, where first-order methods can consistently find approximately worst-case inputs [CW17b; Mad+18].

Spatial loss landscape. To understand why such a difference occurs, we delve deeper into the classifiers to try and understand the failure modes induced by such natural transformations. We find that the loss landscape of classifiers with respect to rotations and translations is highly non-concave and contains many spurious maxima. This is, again, in contrast to the ℓ_p-bounded setting, in which, experimentally, the value of different maxima tend to concentrate well [Mad+18]. Our loss landscape results thus demonstrate that any adversary relying on first order information might

be unable to reliably find misclassifications. Consequently, rigorous evaluation of model robustness in this spatial setting requires techniques that that go beyond what was needed to induce ℓ_p-based adversarial robustness.

Improving spatial robustness. We then develop methods for alleviating these vulnerabilities using insights from our study. As a natural baseline, we augment the training procedure with rotations and translations. While this does largely mitigate the problem on MNIST, additional data augmentation only marginally increases robustness on CIFAR10 and ImageNet. We thus propose two natural methods for further increasing the robustness of these models, based on robust optimization and aggregation of random input transformations. These methods offer significant improvements in classification accuracy against both adaptive and random attackers when compared to both standard models and those trained with additional data augmentation. In particular, on ImageNet, our best model attains a top1 accuracy of **56%** against the strongest adversary, versus **34%** for a standard network with additional data augmentation.

Combining spatial and ℓ_∞-bounded attacks. Finally, we examine the interplay between spatial and ℓ_∞-based perturbations. We observe that robustness to these two classes of input perturbations is largely orthogonal. In particular, pixel-based robustness does not imply spatial robustness, while combining spatial and ℓ_∞-bounded transformations seems to have a *cumulative* effect in reducing classification accuracy. This emphasizes the need to broaden the notions of image similarity in the adversarial examples literature beyond the common ℓ_p-balls.

2.8 Adversarial Rotations and Translations

Recall that in the context of image classification, an *adversarial example* for a given input image x and a classifier C is an image x' that satisfies two properties: (i) on the one hand, the adversarial example x' causes the classifier C to output a different label on x' than on x, i.e., we have $C(x) \neq C(x')$. (ii) On the other hand, the adversarial

example x' is "visually similar" to x.

Clearly, the notion of visual similarity is not precisely defined here. In fact, providing a precise and rigorous definition is extraordinarily difficult as it would require formally capturing the notion of human perception. Consequently, previous work largely settled on the assumption that x' is a valid adversarial example for x if and only if $\|x - x'\|_p \leq \varepsilon$ for some $p \in [0, \infty]$ and ε small enough. This convention is based on the fact that two images are indeed visually similar when they are close enough in some ℓ_p-norm. However, the converse is not necessarily true. A small rotation or translation of an image usually appears visually similar to a human, yet can lead to a large change when measured in an ℓ_p-norm. We aim to expand the range of similarity measures considered in the adversarial examples literature by investigating robustness to small rotations and translations.

Attack methods. Our first goal is to develop sufficiently strong methods for generating adversarial rotations and translations. In the context of pixel-wise ℓ_p-bounded perturbations, the most successful approach for constructing adversarial examples so far has been to employ optimization methods on a suitable loss function [Sze+14; GSS15; CW17b]. Following this approach, we parametrize our attack method with a set of tunable parameters and then optimize over these parameters.

First, we define the exact range of attacks we want to optimize over. For the case of rotation and translation attacks, we wish to find parameters $(\delta u, \delta v, \theta)$ such that rotating the original image by θ degrees around the center and then translating it by $(\delta u, \delta v)$ pixels causes the classifier to make a wrong prediction. Formally, the pixel at position (u, v) is moved to the following position (assuming the point $(0, 0)$ is the center of the image):

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \delta u \\ \delta v \end{bmatrix}.$$

We implement this transformation in a differentiable manner using the spatial trans-

former blocks of [J+15] [15]. In order to handle pixels that are mapped to non-integer coordinates, the transformer units include a differentiable bilinear interpolation routine. Since our loss function is differentiable with respect to the input and the transformation is in turn differentiable with respect to its parameters, we can obtain gradients of the model's loss function w.r.t. the perturbation parameters. This enables us to apply a first-order optimization method to our problem.

By defining the spatial transformation for some x as $T(x; \delta u, \delta v, \theta)$, we construct an adversarial perturbation for x by solving the problem

$$\max_{\delta u, \delta v, \theta} \mathcal{L}(x', y), \quad \text{for } x' = T(x; \delta u, \delta v, \theta) \ , \tag{2.15}$$

where $\mathcal{L}$ is the loss function of the neural network[16], and y is the correct label for x.

We compute the perturbation from Equation 2.15 in three distinct ways:

- **First-Order Method (FO):** Starting from a random choice of parameters, we iteratively take steps in the direction of the gradient of the loss function. This is the direction that locally maximizes the loss of the classifier (as a surrogate for misclassification probability). Since the maximization problem we are optimizing is non-concave, there are no guarantees for global optimality, but the hope is that the local maximum solution closely approximates the global optimum. Note that unlike the ℓ_p-norm case, we are not optimizing in the pixel space but in the latent space of rotation and translation parameters.

- **Grid Search:** We discretize the parameter space and exhaustively examine every possible parametrization of the attack to find one that causes the classifier to give a wrong prediction (if such a parametrization exists). Since our parameter space is low-dimensional enough, this method is computationally feasible (in contrast to a grid search for ℓ_p-based adversaries).

[15] We used the open source implementation found here: `https://github.com/tensorflow/models/tree/master/research/transformer`.

[16] The loss $\mathcal{L}$ of the classifier is a function from images to real numbers that expresses the performance of the network on the particular example x (e.g., the cross-entropy between predicted and correct distributions).

- **Worst-of-k:** We randomly sample k different choices of attack parameters and choose the one on which the model performs worst. As we increase k, this attack interpolates between a random choice and grid search.

We remark that while a first-order attack requires full knowledge of the model to compute the gradient of the loss with respect to the input, the other two attacks do not. They only require the outputs corresponding to chosen inputs, which can be done with only query access to the target model.

2.9 Improving Invariance to Spatial Transformations

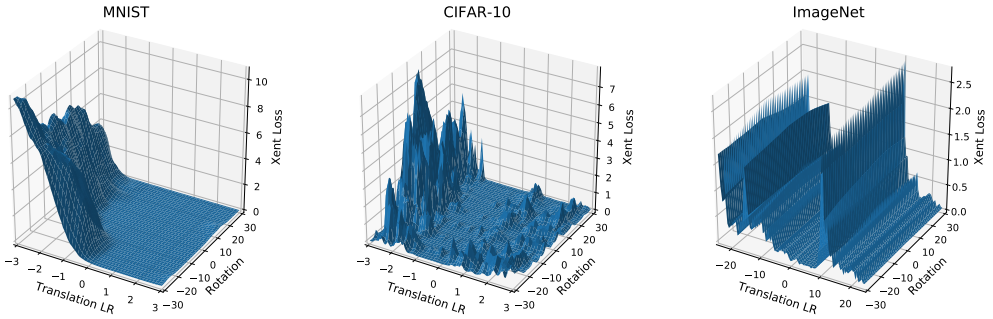


Figure 2-6: Loss landscape of a random example for each dataset when performing left-right translations and rotations. Translations and rotations are restricted to 10% of the image pixels and 30°, respectively. We observe that the landscape is significantly non-concave, rendering first-order methods to generate adversarial example ineffective. Figure A-11 in the appendix shows additional examples.

As we will see in Section 2.10, augmenting the training set with random rotations and translations does improve the robustness of the model against such random transformations. However, data augmentation does not significantly improve the robustness against worst-case attacks and sometimes leads to a drop in accuracy on unperturbed images. To address these issues, we explore two simple baselines that turn out to be surprisingly effective.

2.10 Experiments for Spatial Adversarial Perturbations

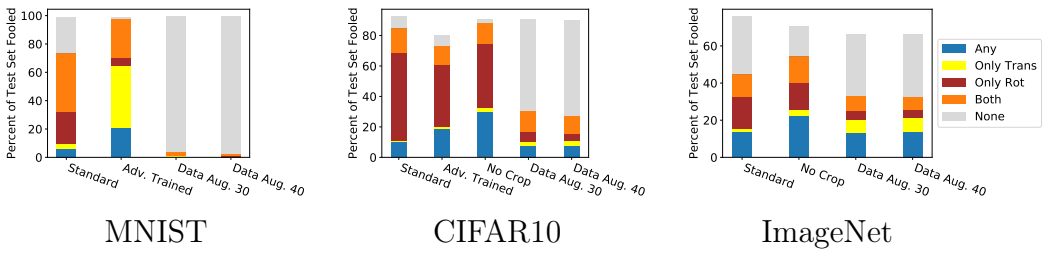


Figure 2-7: Fine-grained dataset analysis. For each model, we visualize what percent of the test set can be fooled via various methods. We compute how many examples can be fooled with either translations or rotations ("any"), how many can be fooled only by one of these, and how many require a combination to be fooled ("both").

We evaluate standard image classifiers for the MNIST [LeC98], CIFAR10 [Kri09] and ImageNet [Rus+15] datasets. In order to determine the extent to which misclassification is caused by insufficient data augmentation during training, we examine various data augmentation methods. We begin with a description of our experimental setup.

Model Architecture. For MNIST, we use a convolutional neural network derived from the TensorFlow Tutorial [Aba+16]. In order to obtain a fully convolutional version of the network, we replace the fully-connected layer by two convolutional layers with 128 and 256 filters each, followed by a global average pooling. For CIFAR10, we consider a standard ResNet [He+16] model with 4 groups of residual layers with filter sizes [16, 16, 32, 64] and 5 residual units each. We use standard and ℓ_∞-adversarially trained models similar to those studied by [Mad+18].[17,18] For ImageNet, we use a ResNet-50 [He+16] architecture implemented in the `tensorpack` repository [Wu+16]. We did not modify the model architectures or training procedures.

[17] `https://github.com/MadryLab/cifar10_challenge`
[18] `https://github.com/MadryLab/mnist_challenge`

Attack Space. In order to maintain the visual similarity of images to the natural ones we restrict the space of allowed perturbations to be relatively small. We consider rotations of at most 30° and translations of at most (roughly) 10% percent of the image size in each direction. This corresponds to 3 pixels for MNIST (image size 28×28) and CIFAR10 (image size 32×32), and 24 pixels for ImageNet (image size 299×299). For grid search attacks, we consider 5 values per translation direction and 31 values for rotations, equally spaced. For first-order attacks, we use 200 steps of projected gradient descent of step size 0.01 times the parameter range. When rotating and translating the images, we fill the empty space with zeros (black pixels).

Data Augmentation. We consider five variants of training for our models.

- Standard training: The standard training procedure for the respective model architecture.
- ℓ_∞-bounded adversarial training: The classifier is trained on ℓ_∞-bounded adversarial examples that are generated with projected gradient descent.
- No random cropping: Standard training for CIFAR-10 and ImageNet includes data augmentation via random crops. We investigate the effect of this data augmentation scheme by also training a model without random crops.
- Random rotations and translations: At each training step, we perform a uniformly random perturbation from the attack space on each training example.
- Random rotations and translations from larger intervals: As before, we perform uniformly random perturbations, but now from a *superset* of the attack space (40°, ± 13% pixels).

2.10.1 Evaluating Model Robustness

We evaluate all models against random and grid search adversaries with rotations and translations considered both separately and together. We report the results in

Table 2.2. We visualize a random subset of successful attacks in Figures A-4, A-5, and A-6 of Appendix A.3.

Despite the high accuracy of standard models on unperturbed examples and their reasonable performance on random perturbations, a grid search can significantly lower the classifiers' accuracy on the test set. For the standard models, accuracy drops from 99% to 26% on MNIST, 93% to 3% on CIFAR10, and 76% to 31% on ImageNet (Top 1 accuracy).

The addition of random rotations and translations during training greatly improves both the random and adversarial accuracy of the classifier for MNIST and CIFAR10, but less so for ImageNet. For the first two datasets, data augmentation increases the accuracy against a grid adversary by 60% to 70%, while the same data augmentation technique adds less than 3% accuracy on ImageNet.

We perform a fine-grained investigation of our findings:

- In Figure 2-7 we examine how many examples can be fooled by (i) rotations only, (ii) translations only, (iii) neither transformation, or (iv) both.

- We visualize the set of fooling angles for a random sample of the rotations-only grid in Figure 2-8 on ImageNet, and provide more examples in the appendix in Figure A-9. We observe that the set of fooling angles is nonconvex and not contiguous.

- To investigate how many transformations are adversarial per image, we analyze the percentage of misclassified grid points for each example in Figure A-10. While the majority of images has only a small number of adversarial transformations, a significant fraction of images is fooled by 20% or more of the transformations.

Padding Experiments. A natural question is whether the reduced accuracy of the models is due to the cropping applied during the transformation. We verify that this is not the case by applying zero and reflection padding to the image datasets. We note that the zero padding creates a "black canvas" version of the dataset, ensuring that no information from the original image is lost after a transformation. We show a

	Model	Nat.	Rand.	Grid	Rand. T.	Grid T.	Rand. R.	Grid R.
MNIST	Standard	99.31%	94.23%	**26.02**%	98.61%	89.80%	95.68%	70.98%
	ℓ_∞-Adv	98.65%	88.02%	**1.20**%	93.72%	34.13%	95.27%	72.03%
	Aug. 30	99.53%	99.35%	**95.79**%	99.47%	98.66%	99.34%	98.23%
	Aug. 40	99.34%	99.31%	**96.95**%	99.39%	98.65%	99.40%	98.49%
	W-10 (30)	99.48%	99.37%	**97.32**%	99.50%	99.01%	99.39%	98.62%
	W-10 (40)	99.42%	99.39%	**97.88**%	99.45%	98.89%	99.36%	98.85%
CIFAR10	Standard	92.62%	60.93%	**2.80**%	88.54%	66.17%	75.36%	24.71%
	No Crop	90.34%	54.64%	**1.86**%	81.95%	46.07%	69.23%	18.34%
	ℓ_∞-Adv	80.21%	58.33%	**6.02**%	78.15%	59.02%	62.85%	20.98%
	Aug. 30	90.02%	90.92%	**58.90**%	91.76%	79.01%	91.14%	76.33%
	Aug. 40	88.83%	91.18%	**61.69**%	91.53%	77.42%	91.10%	76.80%
	W-10 (30)	91.34%	92.35%	**69.17**%	92.43%	83.01%	92.33%	81.82%
	W-10 (40)	91.00%	92.11%	**71.15**%	92.28%	82.15%	92.53%	82.25%
ImageNet	Standard	75.96%	63.39%	**31.42**%	73.24%	60.42%	67.90%	44.98%
	No Crop	70.81%	59.09%	**16.52**%	66.75%	45.17%	62.78%	34.17%
	Aug. 30	65.96%	68.60%	**32.90**%	70.27%	45.72%	69.28%	47.25%
	Aug. 40	66.19%	67.58%	**33.86**%	69.50%	44.60%	68.88%	48.72%
	W-10 (30)	76.14%	73.19%	**52.76**%	74.42%	61.18%	73.74%	61.06%
	W-10 (40)	74.64%	71.36%	**50.23**%	72.86%	59.34%	71.95%	59.23%

Table 2.2: Accuracy of different classifiers against rotation and translation adversaries on MNIST, CIFAR10, and ImageNet. The allowed transformations are translations by (roughly) 10% of the image size and $\pm 30°$ rotations. The attack parameters are chosen through random sampling or grid search with rotations and translations considered both together ("Rand.", "Grid") and separately ("Rand. T." and "Grid T." for transformations, "Rand R." and "Grid R." for rotations). We consider networks that are trained with (i) the respective standard setup, (ii) no data augmentation (if data augmentation is present in standard setup), (iii) with an ℓ_∞ adversary, (iv) with data augmentation corresponding to the attack space (± 3px, $\pm 30°$) and an enlarged space (± 4px, $\pm 40°$), and (v) with worst-of-10 training for both types of augmentations.

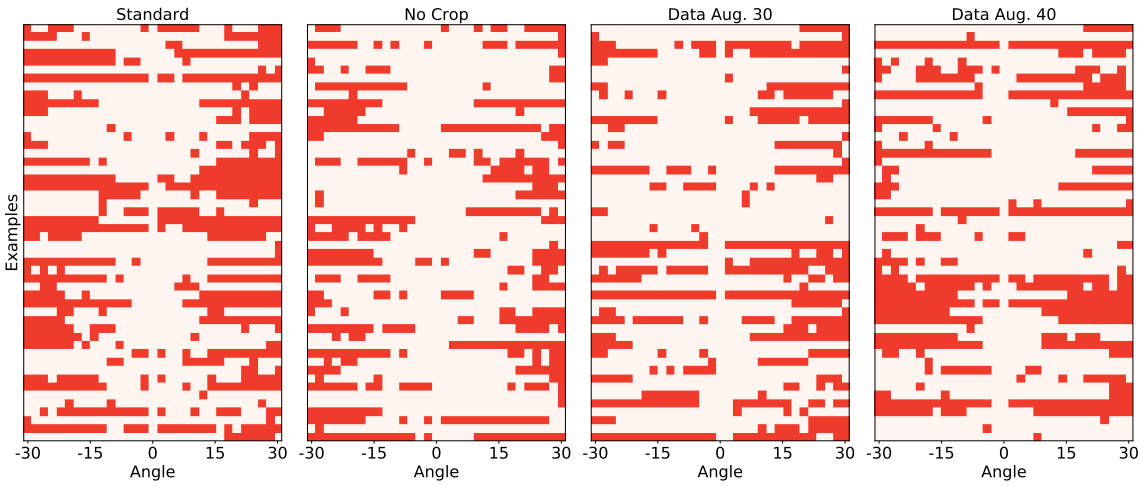


Figure 2-8: Visualizing which angles fool ImageNet classifiers for 50 random examples. For each dataset and model, we visualize one example per row. Red corresponds to *misclassification* of the images. We observe that the angles fooling the models form a highly non-convex set. Figure A-9 in the appendix shows additional examples for CIFAR10 and MNIST.

random set of adversarial examples in this setting in Figure A-7 and a full evaluation in Table A.7. We also provide more details regarding reflection padding in Section A.3.2 and provide an evaluation in Table A.9. All of these are in Appendix A.3.

2.10.2 Comparing Attack Methods

In Table 2.3 we compare different attack methods on various classifiers and datasets. We observe that worst-of-10 is a powerful adversary despite its limited interaction with the target classifier. The first-order adversary performs significantly worse. It fails to approximate the ground-truth accuracy of the models and performs significantly worse than the grid adversary and even the worst-of-10 adversary.

Understanding the Failure of First-Order Methods. The fact that first-order methods fail to reliably find adversarial rotations and translations is in sharp contrast to previous work on ℓ_p-robustness [CW17b; Mad+18]. For ℓ_p-bounded perturbations parametrized directly in pixel space, prior work found the optimization landscape to be well-behaved which allowed first-order methods to consistently find maxima with high loss. In the case of spatial perturbations, we observe that the non-concavity of

	MNIST		CIFAR-10		ImageNet	
	Standard	Aug.	Standard	Aug.	Standard	Aug.
Natural	99.31%	99.53%	92.62%	90.02%	75.96%	65.96%
Worst-of-10	73.32%	98.33%	20.13%	79.92%	47.83%	50.62%
First-Order	79.84%	98.78%	62.69%	85.92%	63.12%	66.05%
Grid	**26.02**%	**95.79**%	**2.80**%	**58.92**%	**31.42**%	**32.90**%

Table 2.3: Comparison of attack methods across datasets and models. Worst-of-10 is very effective and significantly reduces the model accuracy despite the limited interaction. The first-order (FO) adversary performs poorly, despite the large number of steps allowed. We compare standard training to Augmentation ($\pm 3\text{px}, \pm 30°$). For the full table, see Figure A.6 of Appendix A.3.

the problem is a significant barrier for first-order methods. We investigate this issue by visualizing the loss landscape. For a few random examples from the three datasets, we plot the cross-entropy loss of the examples as a function of translation and rotation. Figure 2-6 shows one example for each dataset and additional examples are visualized in Figure A-11 of the appendix. The plots show that the loss landscape is indeed non-concave and contains many local maxima of low value. The low-dimensional problem structure seems to make non-concavity a crucial obstacle. Even for MNIST, where we observe fewer local maxima, the large flat regions prevent first-order methods from finding transformations of high loss.

Relation to Black-Box Attacks. Given its limited interaction with the model, the worst-of-10 adversary achieves a significant reduction in classification accuracy. It performs only 10 *random*, *non-adaptive* queries to the model and is still able to find adversarial examples for a large fraction of the inputs (see Table 2.3). The low query complexity is an important baseline for black-box attacks on neural networks, which recently gained significant interest [Pap+17; Che+17a; Bha+17; Ily+18]. Black-box attacks rely only function evaluations of the target classifier, without additional information such as gradients. The main challenge is to construct an adversarial example from a small number of queries. Our results show that it is possible to find adversarial rotations and translations for a significant fraction of inputs with very few

queries.

Combining Spatial and ℓ_∞-Bounded Perturbations Table 2.2 shows that models trained to be robust to ℓ_∞-bounded perturbations do not achieve higher robustness to spatial perturbations. This provides evidence that the two families of perturbation are orthogonal to each other. We further investigate this possibility by considering a combined adversary that utilizes ℓ_∞-bounded perturbations on top of rotations and translations. The results are shown in Figure A-12. We indeed observe that these combined attacks reduce classification accuracy in an (approximately) additive manner.

2.10.3 Evaluating Our Defense Methods.

As we see in Table 2.2, training with a worst-of-10 adversary significantly increases the spatial robustness of the model, also compared to data augmentation with random transformations. We conjecture that using more reliable methods to compute the worst-case transformations will further improve these results. Unfortunately, increasing the number of random transformations per training example quickly becomes computationally expensive. And as pointed out above, current first-order methods also appear to be insufficient for finding worst-case transformations efficiently.

Our results for majority-based inference are presented in Table A.8 of Appendix A.3. By combining these two defenses, we improve the worst-case performance of the models from 26% to 98% on MNIST, from 3% to 82% on CIFAR10, and from 31% to 56% on ImageNet (Top 1).

2.11 Related Work

Adversarial Robustness. Robust machine learning models are trained with the robust optimization framework [Wal45; GSS15; Mad+18] and were originally proposed in the context of defending against adversarial perturbations [Big+13; Sze+14]. Adversarial robustness has been studied extensively in the context of machine learning

security (see e.g., Carlini and Wagner [CW17a], Athalye et al. [Ath+18], Athalye, Carlini, and Wagner [ACW18], and Papernot et al. [Pap+17]), and as an independent phenomenon (see e.g., Gilmer et al. [Gil+18], Schmidt et al. [Sch+18], Jacobsen et al. [Jac+19], Ilyas et al. [Ily+19], Tsipras et al. [Tsi+19], and Su et al. [Su+18].

Other Explanations for Adversarial Perturbations. Several models for explaining adversarial examples have been proposed in prior work, utilizing ideas ranging from finite-sample overfitting to high-dimensional statistical phenomena [Gil+18; FFF18; For+19; TG16; Sha+19a; MDM18; Sha+19b; GSS15; BPR18]. The key differentiating aspect of our model is that adversarial perturbations arise as *well-generalizing, yet brittle, features*, rather than statistical anomalies or effects of poor statistical concentration. In particular, adversarial vulnerability does not stem from using a specific model class or a specific training method, since standard training on the "robustified" data distribution of Section 2.5.1 leads to robust models. At the same time, as shown in Section 2.5.2, these non-robust features are sufficient to learn a good standard classifier.

Distillation. The experiments performed in Section 2.5.1 can be seen as a form of *distillation*. There is a line of work, known as model distillation [HVD14; Fur+18; BCN06], where the goal is to train a new model to mimic another already trained model. This is typically achieved by adding some regularization terms to the loss in order to encourage the two models to be similar, often replacing training labels with some other target based on the already trained model. While it might be possible to successfully distill a robust model using these methods, our goal was to achieve it by *only* modifying the training set (leaving the training process unchanged), hence demonstrating that adversarial vulnerability is mainly a property of the dataset. Closer to our work is dataset distillation [Wan+18] which considers the problem of reconstructing a classifier from an alternate dataset much smaller than the original training set. This method aims to produce inputs that directly encode the weights of the already trained model by ensuring that the classifier's gradient with respect to

these inputs approximates the desired weights. (As a result, the inputs constructed do not resemble natural inputs.) This approach is orthogonal to our goal since we are not interested in encoding the particular weights into the dataset but rather in imposing a structure to its features.

Adversarial Transferabiliy. In our work, we posit that a potentially natural consequence of the existence of non-robust features is *adversarial transferability* [Pap+17; Liu+17; PMG16]. A recent line of work has considered this phenomenon from a theoretical perspective, confined to simple models, or unbounded perturbations [CRP19; Zou+18]. Tramer et al. [Tra+17] study transferability empirically, by finding *adversarial subspaces*, (orthogonal vectors whose linear combinations are adversarial perturbations). The authors find that there is a significant overlap in the adversarial subspaces between different models, and identify this as a source of transferability. In our work, we provide a potential reason for this overlap—these directions correspond to non-robust features utilized by models in a similar manner.

Universal Adversarial Perturbations Moosavi-Dezfooli et al. [Moo+17] construct perturbations that can cause misclassification when applied to multiple different inputs. More recently, Jetley, Lord, and Torr [JLT18] discover input patterns that are meaningless to humans and can induce misclassification, while at the same time being essential for standard classification. These findings can be naturally cast into our framework by considering these patterns as non-robust features, providing further evidence about their pervasiveness.

Manipulating Dataset Features Ding et al. [Din+19] perform synthetic transformations on the dataset (e.g., image saturation) and study the performance of models on the transformed dataset under standard and robust training. While this can be seen as a method of restricting the features available to the model during training, it is unclear how well these models would perform on the standard test set. Geirhos et al. [Gei+19] aim to quantify the relative dependence of standard models on shape and texture information of the input. They introduce a version of ImageNet where texture

information has been removed and observe an improvement to certain corruptions.

Initial Proposal of Spatial Adversarial Examples The fact that small rotations and translation can fool neural networks on MNIST and CIFAR10 was, to the best of our knowledge, first observed in [FF15]. They compute the minimum transformation required to fool the model and use it as a measure for a quantitative comparison of different architectures and training procedures. The main difference to our work is that we focus on the optimization aspect of the problem. We show that a few random queries usually suffice for a successful attack, while first-order methods are ineffective. Moreover, we go beyond standard data augmentation and evaluate the effectiveness of natural baseline defenses.

Other Work with Spatial Transformations The concurrent work of [KMF18] proposes a different first-order method to evaluate the robustness of classifiers based on geodesic distances on a manifold. This metric is harder to interpret than our parametrized attack space. Moreover, given our findings on the non-concavity of the optimization landscape, it is unclear how close their method is to the ground truth (exhaustive enumeration). While they perform a limited study of defenses (adversarial fine-tuning) using their method, it appears to be less effective than our baseline worst-of-10 training. We attribute this difference to the inherent obstacles first-order methods face in this optimization landscape.

Recently, [Xia+18] and [TB17] observed independently that it is possible to use various spatial transformations to construct adversarial examples for naturally and ℓ_∞-adversarially trained models. The main difference from our work is that we show even very simple transformations (translations and rotations) are sufficient to break a variety of classifiers, while the transformations employed in [Xia+18] and [TB17] are more involved. The transformation in [Xia+18] is based on performing a displacement of individual pixels in the original image constrained to be globally smooth and then optimized for misclassification probability. [TB17] consider an ℓ_∞-bounded pixel-wise perturbation of a version of the original image that has been slightly rotated and in

which a few random pixels have been flipped. Both of these methods require direct access to the attacked model (or a surrogate) to compute (or at least estimate) the gradient of the loss function with respect to the model's input. In contrast, our attacks can be implemented using only a small number of random, non-adaptive inputs.

2.12 Discussion and Conclusion

In this chapter, we first cast the phenomenon of adversarial examples as a natural consequence of the presence of *highly predictive but non-robust features* in standard ML datasets. We provide support for this hypothesis by explicitly disentangling robust and non-robust features in standard datasets, as well as showing that non-robust features alone are sufficient for good generalization. Finally, we study these phenomena in more detail in a theoretical setting where we can rigorously study adversarial vulnerability, robust training, and gradient alignment.

Our findings prompt us to view adversarial examples as a fundamentally *human* phenomenon. In particular, we should not be surprised that classifiers exploit highly predictive features that happen to be non-robust under a human-selected notion of similarity, given such features exist in real-world datasets. In the same manner, from the perspective of interpretability, as long as models rely on these non-robust features, we cannot expect to have model explanations that are both human-meaningful and faithful to the models themselves. Overall, attaining models that are robust and interpretable will require explicitly encoding *human priors* into the training process.

Additionally, we examined the robustness of state-of-the-art image classifiers to translations and rotations. We observed that even a small number of randomly chosen perturbations of the input are sufficient to considerably degrade the classifier's performance. The fact that common neural networks are vulnerable to simple and naturally occurring spatial transformations (and that these transformations can be found easily from just a few random tries) indicates that adversarial robustness should be a concern not only in a fully worst-case security setting. We conjecture that additional techniques need to be incorporated in the architecture and training

procedures of modern classifiers to achieve worst-case spatial robustness. Also, our results underline the need to consider broader notions of similarity than only pixel-wise distances when studying adversarial misclassification attacks. In particular, we view combining the pixel-wise distances with rotations and translations as a next step towards the "right" notion of similarity in the context of images.

Chapter 3

Representations of Adversarially Robust Models

3.1 Introduction

The successes of deep learning models clearly illustrate the utility of learned feature representations. However, these models and their embeddings still exhibit some shortcomings that are at odds with our idealized model of a linear classifier on top of interpretable high-level features. For instance, in Section 2.2, we discussed the pervasive brittleness of deep neural networks [Sze+14; Eng+19a; HD19; Ath+18], particularly to the presence of *adversarial examples* [Big+13; Sze+14]. The existence of adversarial perturbations and the fact that they may correspond to flipping predictive features [Ily+19] suggest that deep neural networks make predictions based on features vastly different from what humans use or recognize. (This message has been also corroborated by several recent works [BB19; Gei+19; JLT18; ZZ19].) In fact, we show a more direct example of such a shortcoming (c.f. Section 3.2), wherein one can construct pairs of images that appear completely different to a human but are nearly identical in terms of their learned feature representations.

With this perspective of adversarial examples, we propose using the robust optimization framework as a tool to enforce (user-specified) *priors* on features that models

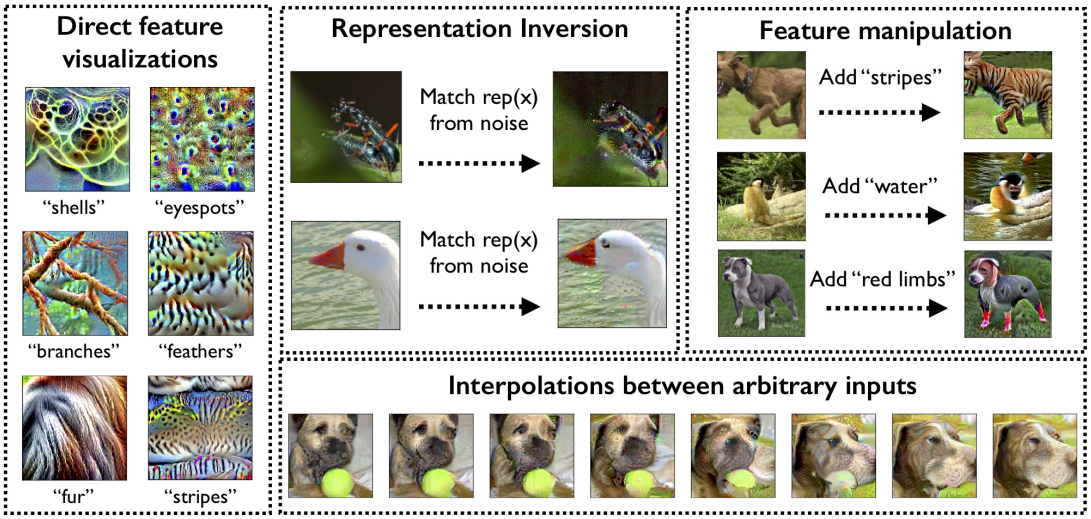


Figure 3-1: Sample images highlighting the properties and applications of "robust representations" studied in this chapter. All of these manipulations use only gradient descent on simple, unregularized, direct functions of the representations of adversarially robust neural networks [GSS15; Mad+18].

should learn (and thus on their learned feature representations). We demonstrate that the resulting learned "robust representations" (the embeddings learned by adversarially robust neural networks [GSS15; Mad+18]) address many of the shortcomings affecting standard learned representations and thereby enable new modes of interaction with inputs via manipulation of salient features. Finally, we demonstrate that by addressing these shortcomings, robust representations enable a new avenue for inspecting model decisions. These findings are summarized below (c.f. Figure 3-1 for an illustration):

- **Representation inversion (Section 3.4.1)**: In stark contrast to standard representations, robust representations are *approximately invertible*—that is, they provide a high-level embedding of the input such that images with similar robust representations are semantically similar, and the salient features of an image are easily recoverable from its robust feature representation. This property also naturally enables feature interpolation between arbitrary inputs.

- **Simple feature visualization (Section 3.4.2)**: Direct maximization of the coordinates of robust representations suffices to visualize easily recognizable features of the model. This is again a significant departure from standard models where

(a) without explicit regularization at visualization time, feature visualization often produces unintelligible results; and (b) even with regularization, visualized features in the representation layer are scarcely human-recognizeable [OMS17].

- **Feature manipulation (Section 3.4.2)**: Through the aforementioned direct feature visualization property, robust representations enable the addition of specific features to images through direct first-order optimization.

- **Model-faithful insights into specific decisions (Section 3.5)**: We can leverage all the properties above in order to gain insight into robust models' decisions on any input via *counterfactual generation*. Similar approaches for standard networks either fail, or require the imposition of human priors at interpretation time, thus sacrificing the veracity of the interpretation with respect to the underlying model.

Beyond model interaction at the representation level, robust optimization and the high-level, meaningful feature space it provides also allows for a number of image synthesis tasks. Once deep learning proved to be successful on image classification, the research frontier expanded into more involved tasks such as image generation [Kar+18; BDS19] and image-to-image translation [Iso+17; Zhu+17]. Much of these new directions, however, were predicated on developing complex, task-specific techniques, often rooted in the generative adversarial network (GAN) framework [Goo+14]. But armed with robust optimization, we demonstrate that basic classification tools *alone* suffice to tackle various image synthesis tasks. These tasks include (cf. Figure 3-2): *generation* (Section 3.7.1), *inpainting* (Section 3.7.2), *image-to-image translation* (Section 3.7.3), *super-resolution* (Section 3.7.4), and *interactive image manipulation* (Section 3.7.5).

Our entire toolkit is based on *a single classifier* (per dataset) and involves performing a simple input manipulation: maximizing predicted class scores with gradient descent. Our approach is thus general purpose and simple to implement and train, while also requiring minimal tuning. To highlight the potential of the core methodology itself, we intentionally employ a generic classification setup (ResNet-50 [He+16] with default hyperparameters) without any additional optimizations (e.g., domain-specific

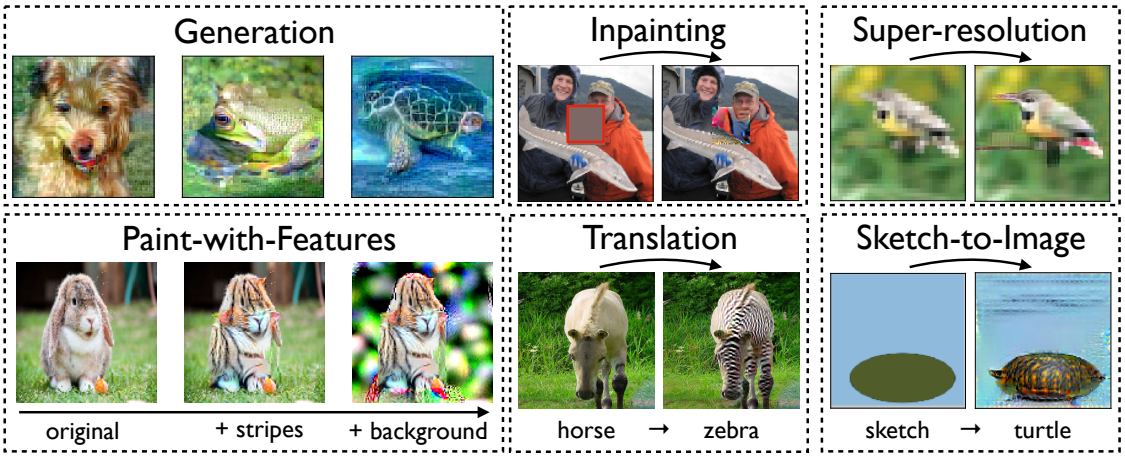


Figure 3-2: Image synthesis and manipulation tasks performed using a *single* (robustly trained) classifier.

priors or regularizers). Moreover, to emphasize the consistency of our approach, we demonstrate performance on *randomly selected* examples from the test set.

Broadly, our results indicate that robust optimization is a promising avenue for learning representations that are more "aligned" with our notion of perception. Moreover, they establish robust classifiers as a powerful primitive for semantic image manipulation, despite them being trained *solely* to perform image classification. Consequently, our findings highlight the the desirability of adversarial robustness as a goal beyond the standard security and reliability context.

3.2 Limitations of standard representations

Following standard convention, for a given deep network we define the *representation* $R(x) \in \mathbb{R}^k$ of a given input $x \in \mathbb{R}^d$ as the activations of the penultimate layer of the network (where usually $k \ll d$). The prediction of the network can thus be viewed as the output of a linear classifier on the representation $R(x)$. We refer to the *distance in representation space* between two inputs (x_1, x_2) as the ℓ_2 distance between their representations $(R(x_1), R(x_2))$, i.e., $\|R(x_1) - R(x_2)\|_2$.

A common aspiration in representation learning is to have that for any pixel-space input x, $R(x)$ is a vector encoding a set of "human-meaningful" features of x [Ben19;

GBC16; BCV13]. These high-level features would be linearly separable with respect to the classification task, allowing the classifier to attain high accuracy.

Running somewhat counter to this intuition, however, we find that it is straightforward to construct pairs of images with nearly identical representations yet drastically different content, as shown in Figure 3-3. Finding such pairs turns out to be as simple as sampling two images $x_1, x_2 \sim \mathcal{D}$, then optimizing one of them to minimize distance in representation space to the other:

$$x_1' = x_1 + \arg\min_{\delta} \|R(x_1 + \delta) - R(x_2)\|_2. \tag{3.1}$$

Indeed, solving objective (3.1) yields images that have similar representations, but share no qualitative resemblance (in fact, x_1' tends to look nearly identical to x_1). An example of such a pair is given in Figure 3-3.

Note that if representations truly provided an encoding of any image into high-level features, finding images with similar representations should necessitate finding images with similar high-level features. Thus, the existence of these image pairs (and similar phenomena observed by prior work [Jac+19]) lays bare a misalignment between the notion of distance induced via the features learned by current deep networks, and the notion of distance as perceived by humans.

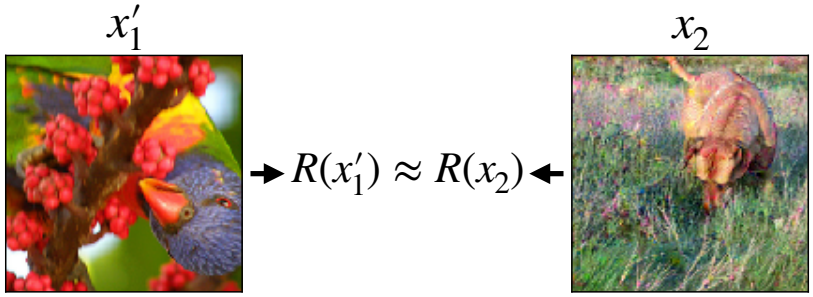


Figure 3-3: A limitation of standard neural network representations: it is straightforward to construct pairs of images (x_1', x_2) that appear completely different yet map to similar representations.

3.3 Adversarial robustness as a prior

Our analysis in Section 3.2 and prior work [Jac+19] prompt the question:

How can we learn better-behaved representations?

In this chapter, we demonstrate that the representations learned by adversarially robust neural networks seem to address many identified limitations of standard representations, and make significant progress towards the broader goal of learning high-level, human-understandable encodings.

Robust optimization as a feature prior. Traditionally, adversarial robustness in the deep learning setting has been explored as a goal predominantly in the context of ML security and reliability [BR18], which we explained in detail in Section 2.2.

Here, we consider an alternative perspective on adversarial robustness—we cast it as a prior on the features that can be learned by a model. Specifically, models trained with objective (2.3) must be *invariant* to a set of perturbations Δ. Thus, selecting Δ to be a set of perturbations that humans are robust to (e.g., small ℓ_p-norm perturbations) results in models that share more invariances with (and thus are encouraged to use similar features to) human perception. Note that incorporating human-selected priors and invariances in this fashion has a long history in the design of ML models—convolutional layers, for instance, were introduced as a means of introducing an invariance to translations of the input [Fuk80].

In what follows, we will explore the effect of the prior induced by adversarial robustness on models' learned representations, and demonstrate that representations learned by adversarially robust models are better behaved, and do in fact seem to use features that are more human-understandable.

3.4 Properties and applications of robust representations

In the previous section, we proposed using *robust optimization* as a way of enforcing user-specified priors during model training. Our goal was to mitigate the issues with standard representations identified in Section 3.2. We now demonstrate that the learned representations resulting from training with this prior indeed exhibit several advantages over standard representations.

Recall that we define a representation $R(\cdot)$ as a function induced by a neural network which maps inputs $x \in \mathbb{R}^n$ to vectors $R(x) \in \mathbb{R}^k$ in the representation layer of that network (the penultimate layer). In what follows, we refer to "standard representations" as the representation functions induced by standard (non-robust) networks, trained with the objective (2.1)—analogously, "robust representations" refer to the representation functions induced by ℓ_2-adversarially robust networks, i.e. networks trained with the objective (2.3) with Δ being the ℓ_2 ball:

$$\theta^*_{robust} = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\delta\|_2 \leq \varepsilon} \mathcal{L}_\theta(x+\delta, y) \right].$$

It is worth noting that despite the value of ε used for training being quite small, we find that robust optimization *globally* affects the behavior of learned representations. As we demonstrate in this section, the benefits of robust representations extend to out-of-distribution inputs and far beyond ε-balls around the training distribution.

Experimental setup. We train robust and standard ResNet-50 [He+16] networks on the Restricted ImageNet [Tsi+19] and ImageNet [Rus+15] datasets. Datasets specifics are in in Appendix A.2.1 and training details are in in Appendices A.2.2. In the main text, we present results for Restricted ImageNet, and link to (nearly identical) results for ImageNet present in the appendices (A.3.13,A.3.15).

Unless explicitly noted otherwise, our optimization method of choice for any objective function will be (projected) gradient descent (PGD), a first-order method

which is known to be highly effective for minimizing neural network-based loss functions for both standard and adversarially robust neural networks [ACW18; Mad+18].

Code for reproducing our results is available at:
`https://github.com/snappymanatee/robust-learned-representations`.

3.4.1 Inverting robust representations

As discussed in Section 3.2, for standard deep networks, given any input x, it is straightforward to find another input that looks entirely different but has nearly the same representation (c.f. Figure 3-3). We noted that this finding runs somewhat counter to the idea that these learned representations effectively capture relevant input features. After all, if the representation function was truly extracting "high-level" features of the input as we conceptualize them, semantically dissimilar images should (by definition) have different representations. We now show that the state of affairs is greatly improved for robust representations.

Robust representations are (approximately) invertible out of the box. We begin by recalling the optimization objective (3.1) used in Section 3.2 to find pairs of images with similar representations, a simple minimization of ℓ_2 distance in representation space from a source image x_1 to a target image x_2:

$$x_1' = x_1 + \arg\min_{\delta} \|R(x_1 + \delta) - R(x_2)\|_2. \tag{3.2}$$

This process can be seen as recovering an image that maps to the desired target representation, and hence is commonly referred to as *representation inversion* [DB16b; MV15; UVL17]. It turns out that in sharp contrast to what we observe for standard models, the images resulting from minimizing (3.2) for robust models are actually *semantically similar* to the original (target) images whose representation is being matched, and this behavior is consistent across multiple samplings of the starting point (source image) x_1 (cf. Figure 3-4).

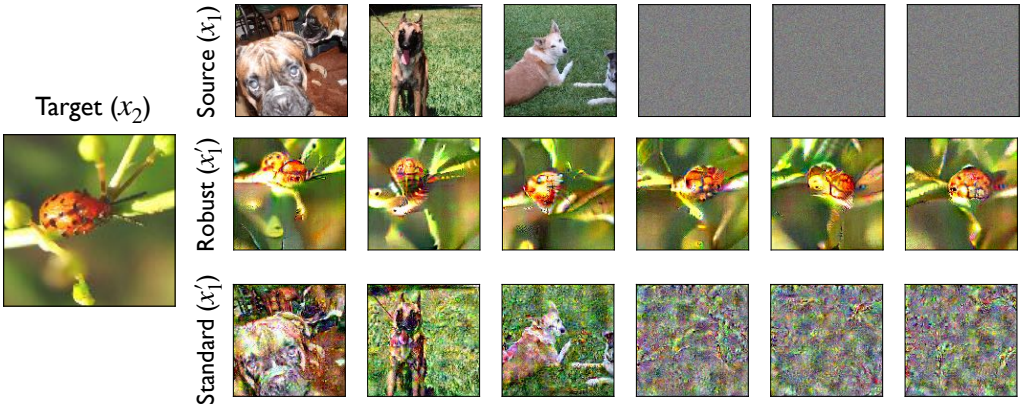


Figure 3-4: Visualization of inputs that are mapped to similar representations by models trained on the Restricted ImageNet dataset. *Target (x_2)* & *Source (x_1)*: random examples image from the test set; *Robust* and *Standard* (x_1'): result of minimizing the objective (3.2) to match (in ℓ_2-distance) the representation of the target image starting from the corresponding source image for (*top*): a robust (adversarially trained) and (*bottom*): a standard model respectively. For the robust model, we observe that the resulting images are perceptually similar to the target image in terms of high-level features (even though they do not match it exactly), while for the standard model they often look more similar to the source image which is the seed for the optimization process. Additional results in Appendix A.3.13, and similar results for ImageNet are in Appendix A.3.13.

Representation proximity seems to entail semantic similarity. In fact, the contrast between the invertibility of standard and robust representations is even stronger. To illustrate this, we will attempt to match the representation of a target image while staying close to the starting image of the optimization in pixel-wise ℓ_2-norm (this is equivalent to putting a norm bound on δ in objective (3.2)). With standard models, we can consistently get close to the target image in representation space, without moving far from the source image x_1. On the other hand, for robust models, we cannot get close to the target representation while staying close to the source image—this is illustrated quantitatively in Figure 3-5. This indicates that for robust models, semantic similarity may in fact be necessary for representation similarity (and is not, for instance, merely an artifact of the local robustness induced by robust optimization).

We also find that even when δ is highly constrained (i.e. when we are forced to stay very close to the source image and thus cannot match the representation of the

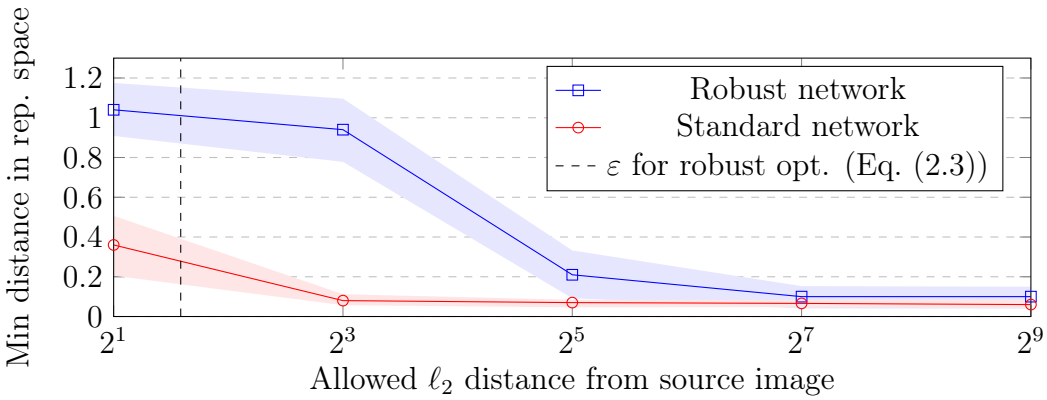


Figure 3-5: Optimizing objective (3.2) with PGD and an ℓ_2-norm constraint around the source image. On the x-axis is the radius of the constraint set, and on the y-axis is the distance in representation space between the minimizer of objective (3.2) within the constraint set and the target image, normalized by the norm of the representation of the target image: i.e., a point (x_i, y_i) on the graph corresponds to $y_i = \min_{\|\delta\|_2 \leq x_i} \|R(x+\delta) - R(x_{targ})\|_2 / \|R(x_{targ})\|_2$. Notably, we are unable to closely match the representation of the target image for the robust network until the norm constraint grows very large, and in particular much larger than the norm of the perturbation that the model is trained to be robust against (ε in objective (2.3)). Shown are 95% confidence intervals over random choice of source and target images.

target well), the solution to the inversion problem still displays some salient features of the target image (c.f. Figure 3-6). Both of these observations suggest that the representations of robust networks function much more like we would expect high-level feature representations to behave.

Inversion of out-of-distribution inputs. We find that the inversion properties uncovered above hold even for out-of-distribution inputs, demonstrating that robust representations capture *general* features as opposed to features only relevant for the specific classification task. In particular, we repeat the inversion experiment (simple minimization of distance in representation space) using images from classes not present in the original dataset used during training (Figure 3-7 right) and structured random patterns (Figure A-22 in Appendix A.3.13): the reconstructed images consistently resemble the targets.

Figure 3-6: A visualization of the final solutions to the optimizing objective (3.2) with PGD when constraining the solution to lie in an ℓ_2 ball around the source image for an adversarially robust neural network. We note that even the radius of the constraint set is small and we cannot match the representation very well, salient features of the target image still arise.

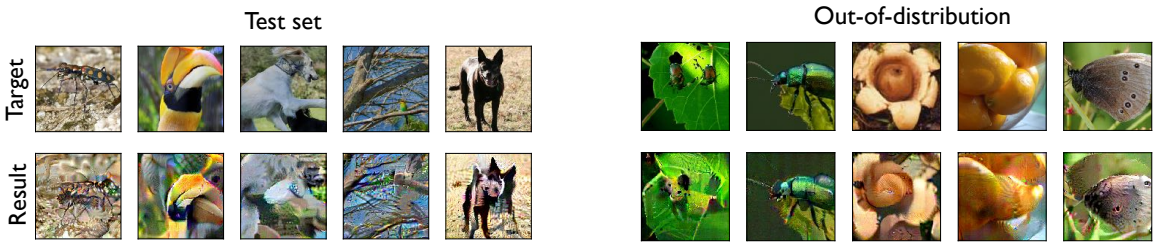


Figure 3-7: Robust representations yield semantically meaningful embeddings. *Target*: random images from the test set (col. 1-5) and from outside of the training distribution (6-10); *Result*: images obtained from optimizing inputs (using Gaussian noise as the source image) to minimize ℓ_2-distance to the representations of the corresponding image in the top row. (More examples appear in Appendix A.3.13.)

Interpolation between arbitrary inputs. Note that this ability to consistently invert representations into corresponding inputs also translates into the ability to *semantically interpolate* between any two inputs. For any two inputs x_1 and x_2, one can (linearly) interpolate between $R(x_1)$ and $R(x_2)$ in representation space, then use the inversion procedure to get images corresponding to the interpolate representations. The resulting inputs interpolate between the two endpoints in a perceptually plausible manner without any of the "ghosting" artifacts present in input-space interpolation. We show examples of this inversion as well as experimental details in Appendix A.2.6.

3.4.2 Direct feature visualization

A common technique for visualizing and understanding the representation function $R(\cdot)$ of a given network is *optimization-based feature visualization* [OMS17], a process

in which we maximize a specific feature (component) in the representation with respect to the input, in order to obtain insight into the role of the feature in classification. Concretely, given some $i \in [k]$ denoting a component of the representation vector, we use gradient descent to find an input x' that maximally activates it, i.e., we solve:

$$x' = x_0 + \arg\max_{\delta} R(x_0 + \delta)_i \tag{3.3}$$

for various starting points x_0 which might be random images from $\mathcal{D}$ or even random noise.

Visualization "fails" for standard networks. For standard networks, optimizing the objective (3.3) often yields unsatisfying results. While we *can* easily find images for which the ith component of $R(\cdot)$ is large (and thus the optimization problem is tractable), these images tends to look meaningless to humans, often resembling the starting point of the optimization. Even when these images are non-trivial, they tend to contain abstract, hard-to-discern patterns (c.f. Figure 3-8 (bottom)). As we discuss later in this section, regularization/post-processing of visualizations does improve this state of affairs, though not very significantly and potentially at the cost of suppressing useful features present in the representation post-hoc.

Robust representations allow for direct visualization of human-recognizable features. For robust representations, however, we find that easily recognizable high-level features emerge from optimizing objective (3.3) directly, *without any regularization or post-processing.* We present the results of this maximization in Figure 3-8 (top): coordinates consistently represent the same concepts across different choice of starting input x_0 (both in and out of distribution). Furthermore, these concepts are not merely an artifact of our visualization process, as they consistently appear in the test-set inputs that most strongly activate their corresponding coordinates (Figure 3-9).

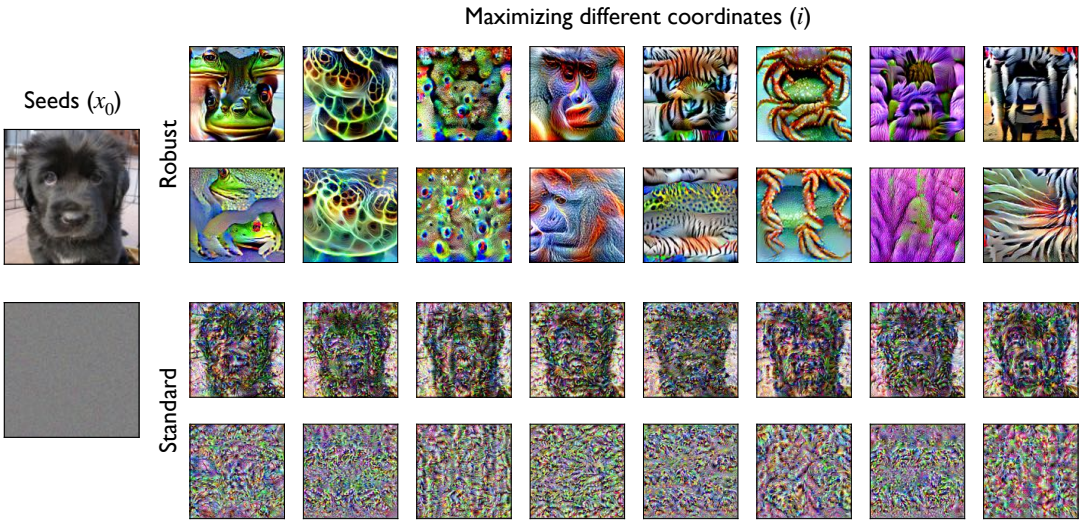


Figure 3-8: Correspondence between image-level patterns and activations learned by standard and robust models on the Restricted ImageNet dataset. Starting from randomly chosen seed inputs (noise/images), we use PGD to find inputs that (locally) maximally activate a given component of the representation vector (cf. Appendix A.2.7 for details). In the left column we have the seed inputs x_0 (selected *randomly*), and in subsequent columns we visualize the result of the optimization (3.3), i.e., x', for different activations, with each row starting from the same (far left) input x_0 for (*top*): a robust (adversarially trained) and (*bottom*): a standard model. Additional visualizations in Appendix A.3.15, and similar results for ImageNet in A.3.15.

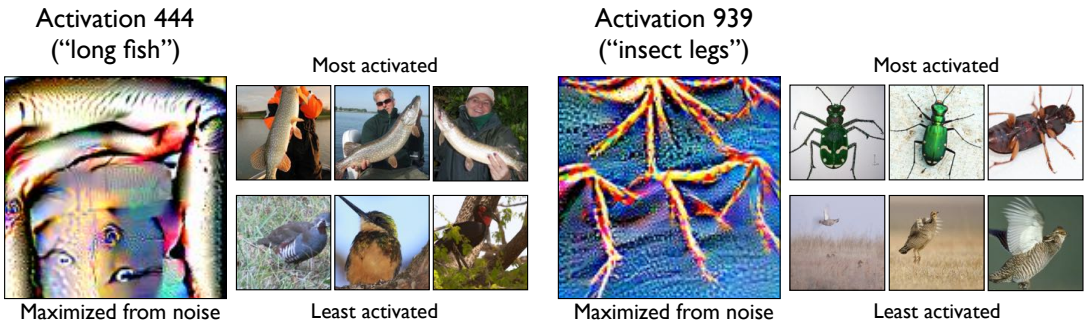
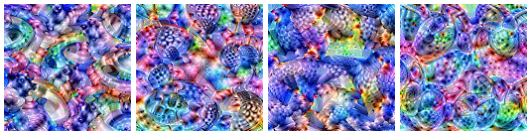


Figure 3-9: Maximizing inputs x' (found by solving (3.3) with x_0 being a gray image) and most or least activating images (from the test set) for two *random* activations of a robust model trained on the Restricted ImageNet dataset. For each activation, we plot the three images from the validation set that had the highest or lowest activation value sorted by the magnitude of the selected activation.

The limitations of regularization for visualization in standard networks. Given that directly optimizing objective (3.3) does not produce human-meaningful images, prior work on visualization usually tries to regularize objective (3.3) through

a variety of methods. These methods include applying random transformations during the optimization process [MOT15; OMS17], restricting the space of possible solutions [NYC15; Ngu+16; Ngu+17], or post-processing the input or gradients [Oyg15; Tyk16]. While regularization does in general produce better results qualitatively, it comes with a few notable disadvantages that are well-recognized in the domain of feature visualization. First, when one introduces prior information about what makes images visually appealing into the optimization process, it becomes difficult to disentangle the effects of the actual model from the effect of the prior information introduced through regularization[1]. Furthermore, while adding regularization does improve the visual quality of the visualizations, the components of the representation still cannot be shown to correspond to any recognizable high-level feature. Indeed, Olah, Mordvintsev, and Schubert [OMS17] note that in the representation layer of a standard GoogLeNet, "Neurons do not seem to correspond to particularly meaningful semantic ideas"—the corresponding feature visualizations are reproduced in Figure 3-10a. We also provide examples of representation-layer visualizations for VGG16 (which we found qualitatively best among modern architectures) regularized with jittering and random rotations in Figure 3-10b. While these visualizations certainly look better qualitatively than their unregularized counterparts in Figure 3-8 (bottom), there remains a significantly large gap in quality and discernability between these regularized visualizations and those of the robust network in Figure 3-8 (top).

(a) Figure reproduced from [OMS17]—a visualization of a few components of the representation layer of GoogLeNet. While regularization (as well as Fourier parameterization and colorspace decorrelation) yields visually appealing results, the visualization does not reveal consistent semantic concepts.



(b) A visualization of the first four components of the representation layer of VGG16 when regularization via random jittering and rotation is applied. Figure produced using the Lucid[a] visualization library.

[a] `https://github.com/tensorflow/lucid/`

[1] In fact, model explanations that enforce priors for purposes of visual appeal have been often found to have little to do with the data or the model itself [Ade+18].

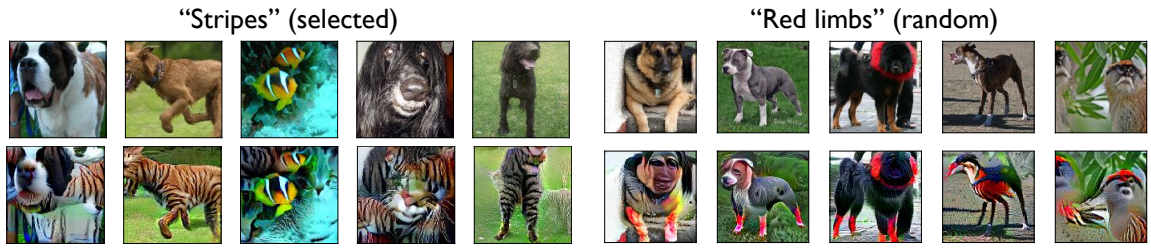


Figure 3-11: Visualization of the results from maximizing a chosen (left) and a *random* (right) representation coordinate starting from *random* images for the Restricted ImageNet dataset. In each figure, the top row has the initial images, and the bottom row has a feature added. Additional examples in Appendix A.3.16.

Natural consequence: feature manipulation

The ability to directly visualize high-level, recognizable features reveals another application of robust representations, which we refer to as *feature manipulation*. Consider the visualization objective (3.3) shown in the previous section. Starting from some original image, optimizing this objective results in the corresponding feature being introduced in a continuous manner. It is hence possible to stop this process relatively early to ensure that the content of the original image is preserved. As a heuristic, we stop the optimization process as soon as the desired feature attains a larger value than all the other coordinates of the representation.

$$\max_{\delta} \begin{cases} R(x+\delta)_i & \text{if } R(x+\delta)_i \leq \max_j R(x+\delta)_j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Intuitively, we maximize the desired feature (the ith coordinate) until it is the highest-magnitude feature in the robust representation. We visualize the result of this process for a variety of input images in Figure 3-11, where *"stripes"* or *"red limbs"* are introduced seamlessly into images without any processing or regularization [2].

[2] We repeat this process with many additional random images and random features in Appendix A.3.16.

3.5 Leveraging robust representations for model-faithful interpretability

The inner mechanisms of standard deep neural networks remain opaque to human observers. In response, significant work has been dedicated to developing *interpretability techniques* that attempt to attribute predictions of deep models to specific parts of the input in a human-meaningful manner [Smi+17; STY17; OMS17; Ola+18].

The standard tradeoff between interpretability and model-faithfulness. The development of tools for interpreting (standard) deep learning models has uncovered a fundamental tension between having explanations that are faithful to the model and explanations that are meaningful to the humans they are presented to. In particular, state-of-the-art methods for interpretability must discard information that models are extremely sensitive to—note, for example, that deep learning models are highly sensitive to adversarial perturbations, but all successful interpretability techniques suppress these perturbations from visualizations due to their lack of human-meaningfulness.

Obtaining human-meaningful interpretations is most often accomplished by enforcing "human priors" *at explanation time* [Ola+18]. These human undoubtedly improve the readability of model explanations, but are often crippled by a lack of faithfulness to the underlying model they are meant to interpret. For example, recent work [NZP18; Ade+18] shows that many interpretation-time priors result in model interpretations that are model- or data-*indepdendent*, and hence inherently unable to explain the predictions of the model.

Interpretation via robust representations. We show that robust representations enable a simple, regularization-free method getting insights into the decisions of robust models. For any input image x, we find the top m (here we use $m = 2$) components of the representation layer that are most activated by x. We then magnify these coordinates in the input by performing the feature addition objective described in Section 3.4.2. The resulting images are similar to the originals, but with the

highest-weight features accentuated. In addition to providing insight about correct classifications (Figure 3-12 right), these images can also suggest *why* certain images are misclassified (Figure 3-12 left). Again, performing a similar analysis for standard model requires incorporating non-trivial human priors in order to make the result human-meaningful (and hence sacrificing faithfulness to the underlying model) [SVZ13; Yos+15; OMS17; Ola+18].

for *interpretation via perturbation* [FV17]. Interpretation via perturbation refers to a model explanation method wherein for an image x, the model explanation consists of a similar image x' (or several such images) along with its (or their) classification(s)—by looking at the qualitative differences between x and x' one can try to discern what features the classifier was using to make its decision[3]. Existing techniques for perturbation-based interpretation suffer from the aforementioned tradeoff and must apply human priors at interpretation time (e.g. using a pretrained generative model, regularization/post-processing, etc.), or else end up with unintelligible model explanations [FV17; Cha+19; DG17].

It turns out, however, that we can exploit robust representations to generating perturbation-based explanations for robust models without enforcing any priors at interpretation time.

In Figure 3-12, accentuating the highest-weight features reveals a natural transformation from the ear of a monkey to the eye of a dog (top), from negative space to the face of a dog (middle), and from the heads of two different fish to the two eyes of a single frog, with a reed transforming into a mouth (bottom). These transformations hint at the most sensitive directions of robust models on misclassified inputs, thus providing insight into model decisions.

[3]This type of interpretation can be seen as somewhat analogous to interpretation via counterfactual generation in the context of causal inference.

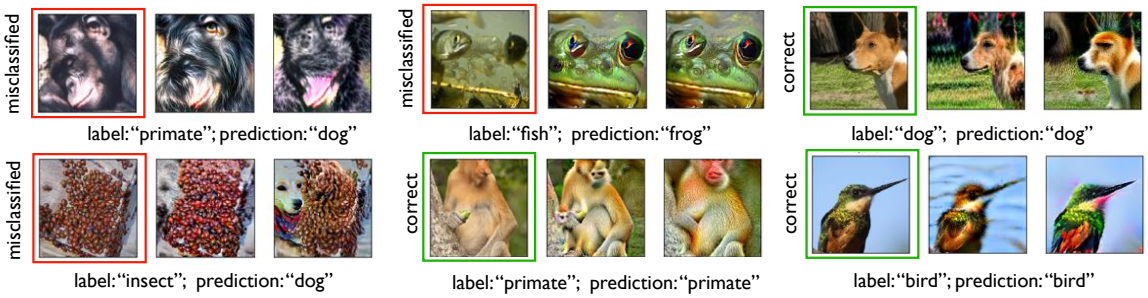


Figure 3-12: Accentuating the highest-weight features for correctly classified (right) and incorrectly (left) classified images. The highest-weight features are accentuated via the "feature addition" mechanism described in Section 3.4.2 (Equation (3.4)). The resulting images appear to show plausible ways in which the images could have been classified/misclassified.

3.6 Robust Models as a Tool for Input Manipulation

In the previous sections, we cast robust optimization as a means of encoding priors into the model, preventing it from relying on imperceptible features of the input [Eng+19b]. Indeed, the findings of Tsipras et al. [Tsi+19] are aligned with this viewpoint—by encouraging the model to be invariant to small perturbations, robust training ensures that changes in the model's predictions correspond to salient input changes.

In fact, it turns out that this phenomenon also emerges when we maximize the probability of a *specific class* (targeted attacks) for a robust model—see Figure 3-13 for an illustration. This indicates that robust models exhibit more human-aligned gradients, and, more importantly, that we can precisely control features in the input just by performing gradient descent on the model output. Previously, performing such manipulations has only been possible with more complex and task-specific techniques [MOT15; RMC16; Iso+17; Zhu+17]. In the rest of this work, we demonstrate that this property of robust models is sufficient to attain good performance on a diverse set of image synthesis tasks.

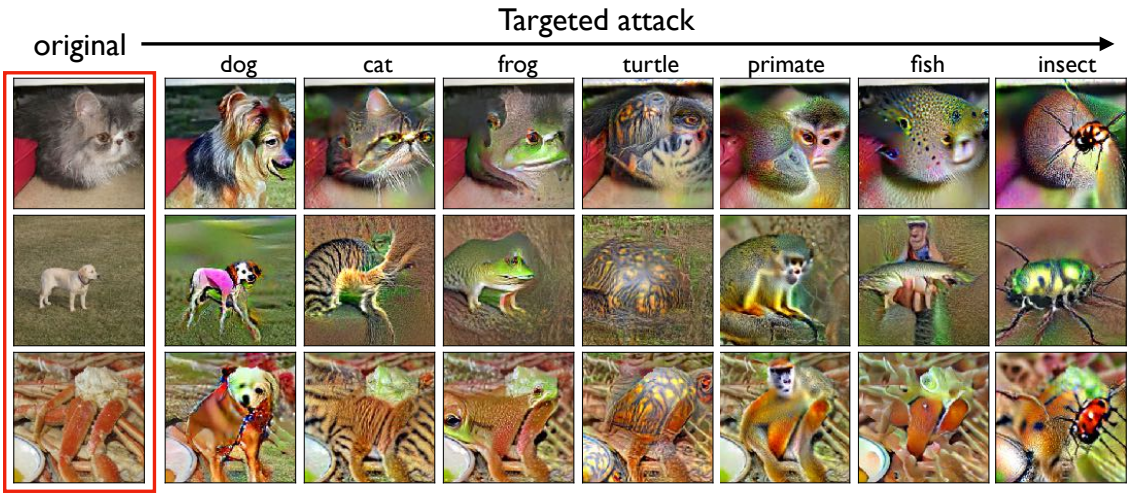


Figure 3-13: Maximizing class scores of a robustly trained classifier. For each original image, we visualize the result of performing targeted projected gradient descent (PGD) toward different classes. The resulting images actually resemble samples of the target class.

3.7 Leveraging Robust Models for Computer Vision Tasks

Deep learning-based methods have recently made significant progress on image synthesis and manipulation tasks, typically by training specifically-crafted models in the GAN framework [Goo+14; ISI17; Zhu+17; Yu+18; BDS19], using priors obtained from deep generative models [Ngu+16; Ngu+17; UVL17; Yeh+17], or leveraging standard classifiers via sophisticated, task-specific methods [MOT15; Oyg15; Tyk16; GEB16]. We discuss additional related work in the following subsections as necessary.

In this section, we outline our methods and results for obtaining competitive performance on these tasks using only robust (feed-forward) classifiers. Our approach is remarkably simple: all the applications are performed using gradient ascent on class scores derived from the same robustly trained classifier. In particular, it does not involve fine-grained tuning, highlighting the potential of robust classifiers as a versatile primitive for sophisticated vision tasks.

3.7.1 Realistic Image Generation

Synthesizing realistic samples for natural data domains (such as images) has been a long standing challenge in computer vision. Given a set of example inputs, we would like to learn a model that can produce novel perceptually-plausible inputs. The development of deep learning-based methods such as autoregressive models [HS97; Gra13; VKK16], auto-encoders [Vin+10; KW15] and flow-based models [DKB14; RM15; DSB17; KD18] has led to significant progress in this domain. More recently, advancements in generative adversarial networks (GANs) [Goo+14] have made it possible to generate high-quality images for challenging datasets [Zha+18a; Kar+18; BDS19]. Many of these methods, however, can be tricky to train and properly tune. They are also fairly computationally intensive, and often require fine-grained performance optimizations.

In contrast, we demonstrate that robust classifiers, without any special training or auxiliary networks, can be a powerful tool for synthesizing realistic natural images. At a high level, our generation procedure is based on maximizing the class score of the desired class using a robust model. The purpose of this maximization is to add relevant and semantically meaningful features of that class to a given input image. This approach has been previously used on standard models to perform class visualization—synthesizing prototypical inputs of each class—in combination with domain-specific input priors (either hand-crafted [NYC15] and learned [Ngu+16; Ngu+17]) or regularizers [SVZ13; MOT15; Oyg15; Tyk16].

As the process of class score maximization is deterministic, generating a diverse set of samples requires a random seed as the starting point of the maximization process. Formally, to generate a sample of class y, we sample a seed and minimize the loss $\mathcal{L}$ of label y

$$x = \underset{\|x'-x_0\|_2 \leq \varepsilon}{\arg\min}\ \mathcal{L}(x', y), \qquad x_0 \sim \mathcal{G}_y,$$

for some class-conditional seed distribution $\mathcal{G}_y$, using projected gradient descent (PGD) (experimental details can be found in Appendix A.2). Ideally, samples from $\mathcal{G}_y$ should be diverse and statistically similar to the data distribution. Here, we use a simple

(but already sufficient) choice for $\mathcal{G}_y$—a multivariate normal distribution fit to the empirical class-conditional distribution

$$\mathcal{G}_y := \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y), \quad \text{where } \boldsymbol{\mu}_y = \mathbb{E}_{x \sim \mathcal{D}_y}[x],\ \boldsymbol{\Sigma} = \mathbb{E}_{x \sim \mathcal{D}_y}[(x - \boldsymbol{\mu}_y)^\top (x - \boldsymbol{\mu}_y)],$$

and $\mathcal{D}_y$ is the distribution of natural inputs conditioned on the label y. We visualize example seeds from these multivariate Gaussians in Figure A-40.

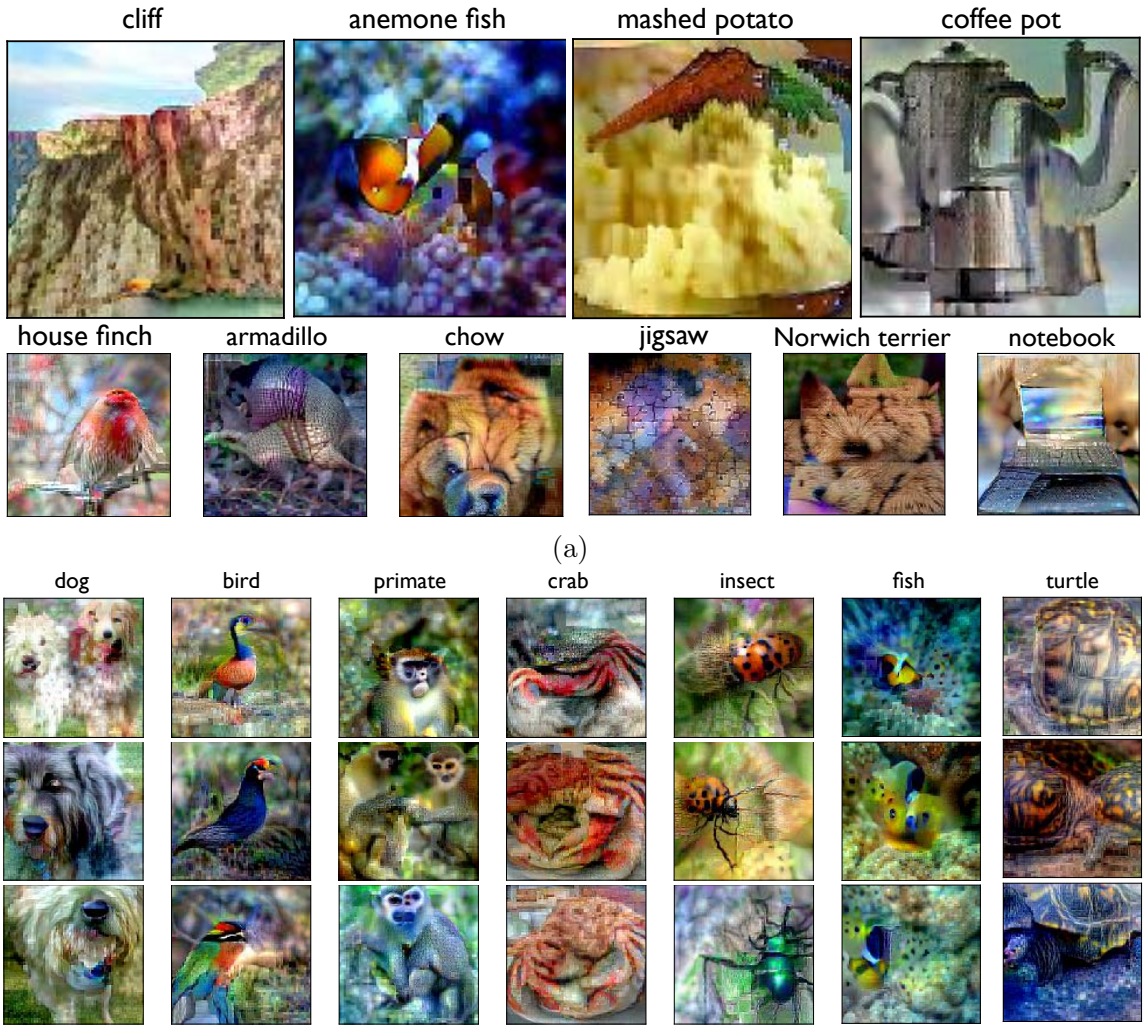


Figure 3-14: *Random* samples (of resolution 224×224) produced using a robustly trained classifier. We show: (a) samples from several (random) classes of the ImageNet dataset and (b) multiple samples from a few random classes of the restricted ImageNet dataset (to illustrate diversity). See Figures A-36, A-37, A-38, and A-39 of Appendix A.3 for additional samples.

This approach enables us to perform *conditional* image synthesis given any target class. Samples (at resolution 224×224) produced by our method are shown in Figure 3-14 (also see Appendix A.3). The resulting images are diverse and realistic, despite the fact that they are generated using targeted PGD on off-the-shelf robust models without any additional optimizations. [4]

Different seed distributions. It is worth noting that there is significant room for improvement in designing the distribution $\mathcal{G}_y$. One way to synthesize better samples would be to use a richer distribution—for instance, mixtures of Gaussians per class to better capture multiple data modes. Also, in contrast to many existing approaches, we are not limited to a single seed distribution, and we could even utilize other methods (such as procedural generation) to customize seeds with specific structure or color, and then maximize class scores to produce realistic samples (e.g., see Section 3.7.5).

Evaluating Sample Quality. Inception Score (IS) [Sal+16] is a popular metric for evaluating the quality of generated image data. Table 3.1 presents the IS of samples generated using a robust classifier.

Dataset	Train Data	BigGAN	WGAN-GP	Our approach
CIFAR-10	11.2 ± 0.2	9.22	8.4 ± 0.1	7.5 ± 0.1
ImageNet[5]	331.9 ± 4.9	233.1 ± 1	11.6	259.0 ± 4

Table 3.1: Inception Scores (IS) for samples generated using robustly trained classifiers compared to state-of-the-art generation approaches [Gul+17; SSA18; BDS19] (cf. Appendix A.2.10 for details).

We find that our approach improves over state-of-the-art (BigGAN [BDS19]) in terms of Inception Score on the ImageNet dataset, yet, at the same time, the Fréchet Inception Distance (FID) [Heu+17] is worse (36.0 versus 7.4). These results can be explained by the fact that, on one hand, our samples are essentially adversarial examples (which are known to transfer across models [Sze+14]) and thus are likely

[4]Interestingly, the robust model used to generate these high-quality ImageNet samples is only 45% accurate, yet has a sufficiently rich representation to synthesize semantic features for 1000 classes.

[5]For ImageNet, there is a difference in resolution between BigGAN samples (256×256), SAGAN (128×128) and our approach (224×224). BigGAN attains IS of 166.5. at 128×128 resolution.

to induce highly confident predictions that IS is designed to pick up. On the other hand, GANs are explicitly trained to produce samples that are indistinguishable from true data with respect to a discriminator, and hence are likely to have a better (lower) FID.

3.7.2 Inpainting

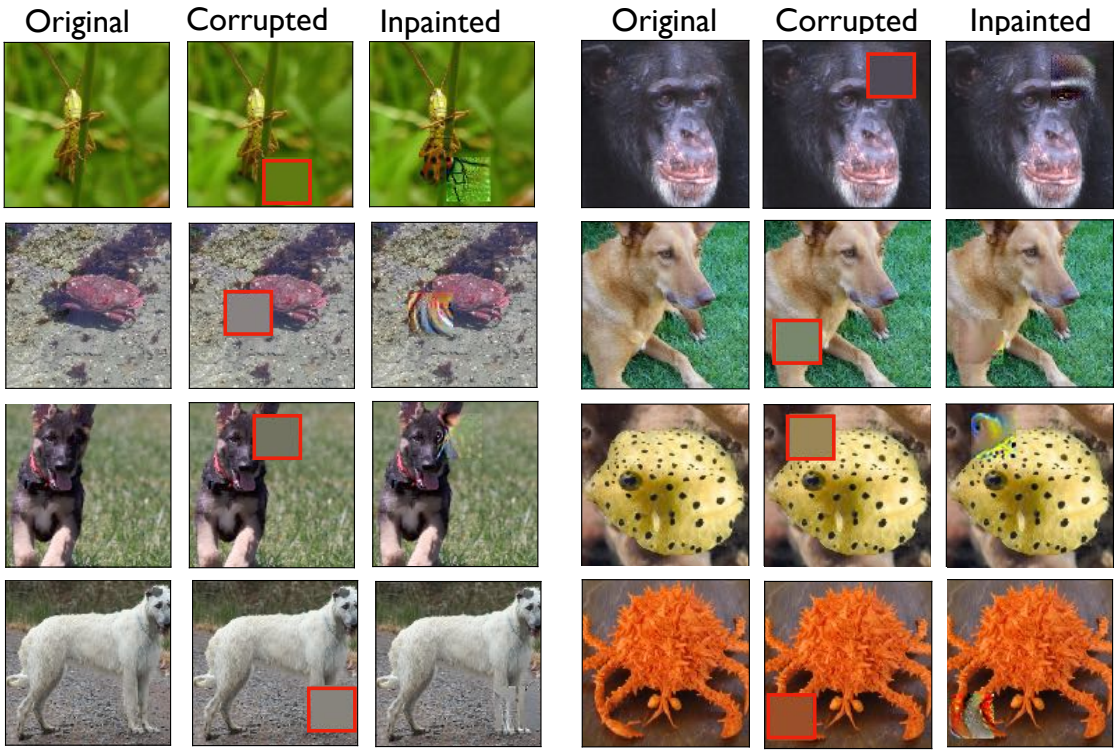
Image inpainting is the task of recovering images with large corrupted regions [EL99; Ber+00; HE07]. Given an image x, corrupted in a region corresponding to a binary mask $m \in \{0,1\}^d$, the goal of inpainting is to recover the missing pixels in a manner that is perceptually plausible with respect to the rest of the image. We find that simple feed-forward classifiers, when robustly trained, can be a powerful tool for such image reconstruction tasks.

From our perspective, the goal is to use robust models to restore missing features of the image. To this end, we will optimize the image to maximize the score of the underlying true class, while also forcing it to be consistent with the original in the uncorrupted regions. Concretely, given a robust classifier trained on uncorrupted data, and a corrupted image x with label y, we solve

$$x_I = \arg\min_{x'} \mathcal{L}(x', y) + \lambda ||(x - x') \odot (1 - m)||_2 \tag{3.5}$$

where $\mathcal{L}$ is the cross-entropy loss, $\odot$ denotes element-wise multiplication, and λ is an appropriately chosen constant. Note that while we require knowing the underlying label y for the input, it can typically be accurately predicted by the classifier itself given the corrupted image.

In Figure 3-15, we show sample reconstructions obtained by optimizing (3.5) using PGD (cf. Appendix A.2 for details). We can observe that these reconstructions look remarkably similar to the uncorrupted images in terms of semantic content. Interestingly, even when this approach fails (reconstructions differ from the original), the resulting images do tend to be perceptually plausible to a human, as shown in Appendix Figure A-35.



(a) *random* samples (b) select samples

Figure 3-15: Image inpainting using robust models – *left:* original, *middle:* corrupted and *right:* inpainted samples. To recover missing regions, we use PGD to maximize the class score predicted for the image while penalizing changes to the uncorrupted regions.

3.7.3 Image-to-Image Translation

As discussed in Section 3.6, robust models provide a mechanism for transforming inputs between classes. In computer vision literature, this would be an instance of *image-to-image translation*, where the goal is to translate an image from a source to a target domain in a semantic manner [Her+01].

In this section, we demonstrate that robust classifiers give rise to a new methodology for performing such image-to-image translations. The key is to (robustly) train a classifier to distinguish between the source and target domain. Conceptually, such a classifier will extract salient characteristics of each domain in order to make accurate predictions. We can then translate an input from the source domain by directly maximizing the predicted score of the target domain.

In Figure 3-16, we provide sample translations produced by our approach using robust models—each trained only on the source and target domains for the Horse ↔ Zebra, Apple ↔ Orange, and Summer ↔ Winter datasets [Zhu+17] respectively. (For completeness, we present in Appendix A.3 Figure A-33 results corresponding to using a classifier trained on the complete ImageNet dataset.) In general, we find that this procedure yields meaningful translations by directly modifying characteristics of the image that are strongly tied to the corresponding domain (e.g., color, texture, stripes).

(a) *random* samples (b) select samples

Figure 3-16: Image-to-image translation on the Horse ↔ Zebra, Apple ↔ Orange, and Summer ↔ Winter datasets [Zhu+17] using PGD on the input of an ℓ_2-robust model trained on that dataset. See Appendix A.2 for experimental details and Figure A-32 for additional input-output pairs.

Note that, in order to manipulate such features, the model must have learned them in the first place—for example, we want models to distinguish between horses and zebras based on salient features such as stripes. For overly simple tasks, models might extract little salient information (e.g., by relying on backgrounds instead of objects[6]) in which case our approach would not lead to meaningful translations. Nevertheless, this not a fundamental barrier and can be addressed by training on richer, more

[6]In fact, we encountered such an issue with ℓ_∞-robust classifiers for horses and zebras (Figure A-34). Note that generative approaches also face similar issues, where the background is transformed instead of the objects [Zhu+17].

challenging datasets. From this perspective, scaling to larger datasets (which can be difficult for state-of-the-art methods such as GANs) is actually easy and advantageous for our approach.

Unpaired datasets. Datasets for translation tasks often comprise source-target domain pairs [Iso+17]. For such datasets, the task can be straightforwardly cast into a supervised learning framework. In contrast, our method operates in the *unpaired* setting, where samples from the source and target domain are provided without an explicit pairing [Zhu+17]. This is due to the fact that our method only requires a classifier capable of distinguishing between the source and target domains.

3.7.4 Super-Resolution

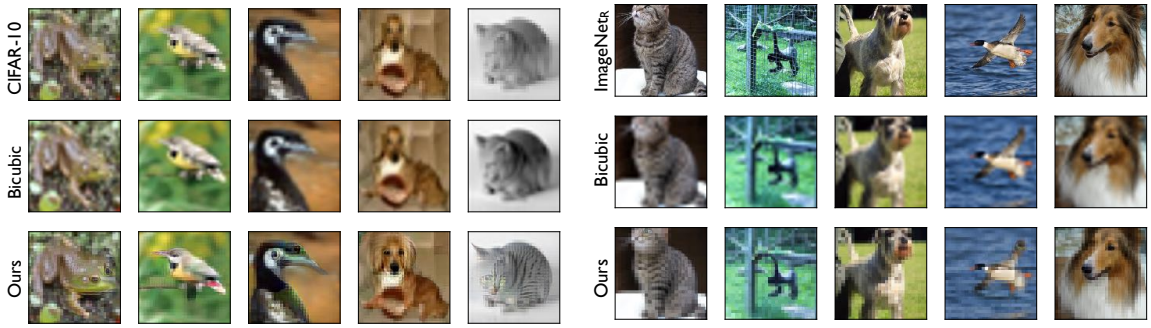
Super-resolution refers to the task of recovering high-resolution images given their low resolution version [DFE07; BSH12]. While this goal is underspecified, our aim is to produce a high-resolution image that is consistent with the input and plausible to a human.

In order to adapt our framework to this problem, we cast super-resolution as the task of accentuating the salient features of low-resolution images. This can be achieved by maximizing the score predicted by a robust classifier (trained on the original high-resolution dataset) for the underlying class. At the same time, to ensure that the structure and high-level content is preserved, we penalize large deviations from the original low-resolution image. Formally, given a robust classifier and a low-resolution image x_L belonging to class y, we use PGD to solve

$$\hat{x}_H = \underset{||x' - \uparrow(x_L)|| < \varepsilon}{\arg\min} \mathcal{L}(x', y) \tag{3.6}$$

where $\uparrow(\cdot)$ denotes the up-sampling operation based on nearest neighbors, and ε is a small constant.

We use this approach to upsample *random* 32×32 CIFAR-10 images to full ImageNet size (224×224)—cf. Figure 3-17a. For comparison, we also show upsampled



(a) 7x super-resolution on CIFAR-10 (b) 8x super-resolution on restricted ImageNet

Figure 3-17: Comparing approaches for super-resolution. *Top: random* samples from the test set; *middle:* upsampling using bicubic interpolation; and *bottom:* super-resolution using robust models. We obtain semantically meaningful reconstructions that are especially sharp in regions that contain class-relevant information.

images obtained from bicubic interpolation. In Figure 3-17b, we visualize the results for super-resolution on *random* 8-fold down-sampled images from the restricted ImageNet dataset. Since in the latter case we have access to ground truth high-resolution images (actual dataset samples), we can compute the Peak Signal-to-Noise Ratio (PSNR) of the reconstructions. Over the Restricted ImageNet test set, our approach yields a PSNR of 21.53 (95% CI [21.49, 21.58]) compared to 21.30 (95% CI [21.25, 21.35]) from bicubic interpolation. In general, our approach produces high-resolution samples that are substantially sharper, particularly in regions of the image that contain salient class information.

Note that the pixelation of the resulting images can be attributed to using a very crude upsampling of the original, low-resolution image as a starting point for our optimization. Combining this method with a more sophisticated initialization scheme (e.g., bicubic interpolation) is likely to yield better overall results.

3.7.5 Interactive Image Manipulation

Recent work has explored building deep learning–based interactive tools for image synthesis and manipulation. For example, GANs have been used to transform simple sketches [CH18; Par+19] into realistic images. In fact, recent work has pushed this one step further by building a tool that allows object-level composition of scenes using

GANs [Bau+19a]. In this section, we show how our framework can be used to enable similar artistic applications.

Sketch-to-image. By performing PGD to maximize the probability of a chosen target class, we can use robust models to convert hand-drawn sketches to natural images. The resulting images (Figure 3-18) appear realistic and contain fine-grained characteristics of the corresponding class.

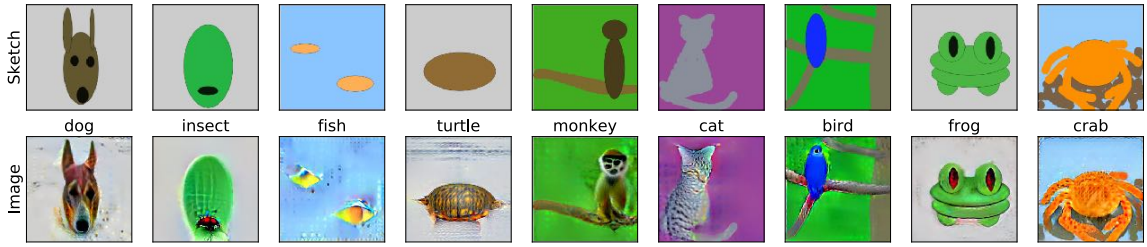


Figure 3-18: Sketch-to-image using robust model gradients. *Top:* manually drawn sketches of animals; and *bottom:* result of performing PGD towards a chosen class. The resulting images appear realistic looking while preserving key characteristics of the original sketches[7].

Feature Painting. Generative model–based paint applications often allow the user to control more fine-grained features, as opposed to just the overall class. We now show that we can perform similar feature manipulation through a minor modification to our basic primitive of class score maximization. Our methodology is based on an observation of Engstrom et al. [Eng+19b], wherein manipulating individual activations within representations[8] of a robust model actually results in consistent and meaningful changes to high-level image features (e.g., adding stripes to objects). We can thus build a tool to paint specific features onto images by maximizing individual activations directly, instead of just the class scores.

Concretely, given an image x, if we want to add a single feature corresponding to component f of the representation vector $R(x)$ in the region corresponding to a

[7]Sketches were produced by a graduate student without any training in arts.

[8]We refer to the pre-final layer of a network as the representation layer. Then, the network prediction can simply be viewed as the output of a linear classifier on the representation.

binary mask m, we simply apply PGD to solve

$$x_I = \arg\max_{x'} R(x')_f - \lambda_P ||(x - x') \odot (1 - m)||. \tag{3.7}$$

In Figure 3-19, we demonstrate progressive addition of features at various levels of

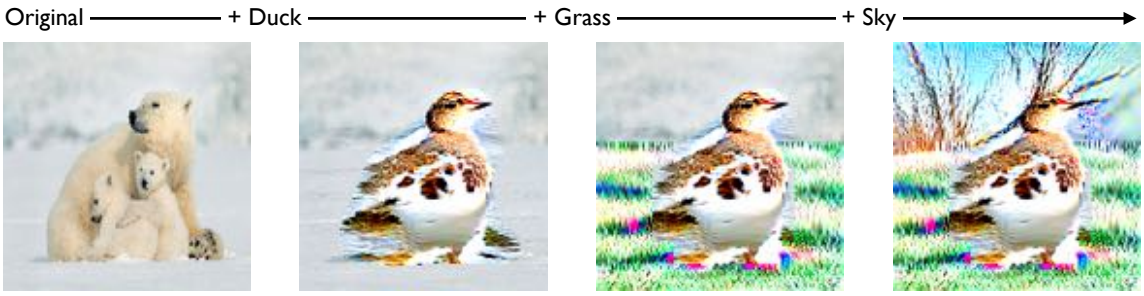


Figure 3-19: Paint-with-features using a robust model—we present a sequence of images obtained by successively adding specific features to select regions of the image by solving (3.7).

granularity (e.g., grass or sky) to selected regions of the input image. We can observe that such direct maximization of individual activations gives rise to a versatile paint tool.

3.8 Related Work

Inverting representations. Previous methods for inverting learned representations typically either solve an optimization problem similar to (3.1) while imposing a "natural image" prior on the input [MV15; Yos+15; UVL17] or train a separate model to perform the inversion [KW15; DB16b; DB16a]. Note that since these methods introduce priors or additional components into the inversion process, their results are not fully faithful to the model. In an orthogonal direction, it is possible to construct models that are analytically invertible by construction [DKB14; DSB17; JSO18; Beh+18]. However, the representations learned by these models do not seem to be perceptually meaningful (for instance, interpolating between points in the representation space does not lead to perceptual input space interpolations [JSO18]). Another notable distinction between the inversions shown here and invertible networks is that the latter are an exactly

invertible map from $\mathbb{R}^d \to \mathbb{R}^d$, while the former shows that we can approximately recover the original input in $\mathbb{R}^d$ from a representation in $\mathbb{R}^k$ for $k \ll d$.

Feature visualization. Typical methods for visualizing features or classes learned by deep networks follow an optimization-based approach, optimizing objectives similar to objective (3.3). Since this optimization does not lead to meaningful visualizations directly, these methods incorporate domain-specific input priors (either hand-crafted [NYC15] or learned [Ngu+16; Ngu+17]) and regularizers [SVZ13; MOT15; Oyg15; Yos+15; Tyk16; OMS17] to produce human-discernible visualizations. The key difference of our work is that we avoid the use of such priors or regularizers altogether, hence producing visualizations that are fully based on the model itself without introducing any additional bias.

Semantic feature manipulation. The latent space of generative adversarial networks (GANs) [Goo+14] tends to allow for "semantic feature arithmetic" [RMC16; Lar+16] (similar to that in word2vec embeddings [Mik+13]) where one can manipulate salient input features using latent space manipulations. In a similar vein, one can utilize an image-to-image translation framework to perform such manipulation (e.g. transforming horses to zebras), although this requires a task-specific dataset and model [Zhu+17]. Somewhat orthogonally, it is possible to utilize the deep representations of *standard* models to perform semantic feature manipulations; however such methods tend to either only perform well on datasets where the inputs are center-aligned [Upc+17], or are restricted to a small set of manipulations [GEB16].

3.9 Discussion and Conclusion

Our findings show that adversarially training yields learned representations that align much more closely with our idealized view of neural network embeddings as extractors of human-meaningful, high-level features. That is, introducing the robustness prior at training time overcomes many of the challenges identified for standard representations. We demonstrate that robust model are approximately invertible, and moving towards

an image in representation space seems to entail recovering salient features of that image in pixel space. Furthermore, we show that robust representations can be directly visualized with first-order methods without the need for post-processing or regularization, and also yield much more human-understandable features than standard models (even when they are visualized with regularization). These two properties (inversion and direct feature visualization), in addition to serving as illustrations of the benefits of robust representations, also enable direct modes of input manipulation (interpolation and feature manipulation, respectively).

Finally, by using robust models, we are able to leverage the basic classification framework to perform a wide range of image synthesis tasks. We then show how this insight gives rise to a versatile toolkit that is simple, reliable, and straightforward to extend to other large-scale datasets. This is in stark contrast to state-of-the-art approaches [Goo+14; Kar+18; BDS19] which typically rely on architectural, algorithmic, and task-specific optimizations to succeed at scale [Sal+16; Das+18; Miy+18]. In fact, unlike these approaches, our methods actually *benefit* from scaling to more complex datasets—whenever the underlying classification task is rich and challenging, the classifier is likely to learn more fine-grained features.

We also note that we refrain from using extensive tuning or task-specific optimizations. This is intended to demonstrate the potential of our core framework itself, rather than to exactly match/outperform the state of the art. We fully expect that better training methods, improved notions of robustness, and domain knowledge will yield even better results.

More broadly, our findings suggest two high-level ideas for future works: First, adversarial robustness may be a property desirable beyond security and reliability contexts. And second, we chose to use robust optimization as a framework to enforce feature priors, but there are many more priors containing invariances in the human visual system. We believe that further exploring this paradigm will lead to models that are significantly more human-aligned while enabling a wide range of interactions.

Chapter 4

Data Poisoning by Introducing Features

4.1 Introduction

In the previous chapters, we discussed the susceptibility of deep learning models to adversarial examples as well as the benefits obtained after robust training to protect against such threats. Now, we turn to an orthogonal, yet also important, concern in the context the security of neural nets, specifically their vulnerability to manipulation of their training sets. Such networks are often fairly data hungry, resulting in training on data that could not be properly vetted. Consequently, any gathered data might have been manipulated by a malicious adversary and cannot necessarily be trusted. One well-studied setting for such training set attacks is *data poisoning* [BNL12; Xia+15; MZ15; SKL17; KL17]. Here, the adversary injects a small number of corrupted training examples, with a goal of degrading the model's generalization accuracy.

However, state of the art poisoning attacks fail to significantly degrade the test accuracy of deep neural networks, and it is still unknown why deep nets are so resistant to traditional data poisoning. As such, we instead focus on different, more sophisticated forms of poisoning. These new threat models involve the purposeful addition of malicious features into the training set so as to corrupt the data.

To the best of our knowledge, the first successful poisoning threat models on deep neural networks were the so-called *backdoor* attacks [GDG17; Che+17b; Adi+18]. Rather than causing the model's test accuracy to degrade, the adversary's goal is for the network to misclassify the test inputs when the data point has been altered by the adversary's choice of perturbation. This is particularly insidious since the network correctly classifies typical test examples, and so it can be hard to detect if the dataset has been corrupted.

Oftentimes, these attacks are straightforward to implement. Many simply involve adding a small number of corrupted examples from a chosen attack class, mislabelled with a chosen target class. This simple change to the training set is then enough to achieve the desired results of a network that correctly classifies clean test inputs while also misclassifying backdoored test inputs. Despite their apparent simplicity, though, no effective defenses to these attacks are known.

4.1.1 Backdoor Defense

Our contribution is in demonstrating a new property of backdoor attacks. Specifically, we show that these attacks tend to leave behind a detectable trace in the spectrum of the covariance of a feature representation learned by the neural network. We call this "trace" a *spectral signature*. We demonstrate that one can use this signature to identify and remove corrupted inputs. On CIFAR-10, which contains 5000 images for each of 10 labels, we show that with as few as 250 corrupted training examples, the model can be trained to misclassify more than 90% of test examples modified to contain the backdoor. In our experiments, we are able to use spectral signatures to reliably remove many—in fact, often all—of the corrupted training examples, reducing the misclassification rate on backdoored test points to within 1% of the rate achieved by a standard network trained on a clean training set. Moreover, we provide some intuition for why one might expect an overparameterized neural network to naturally install a backdoor, and why this also lends itself to the presence of a spectral signature. Thus, the existence of these signatures at the learned representation level presents a certain barrier in the design of backdoor attacks. To create an undetectable attack

would require either ruling out the existence of spectral signatures or arguing that backpropogation will never create them. We view this as a first step towards developing comprehensive defenses against backdoor attacks.

Our notion of spectral signatures draws from a new connection to recent techniques developed for robust statistics [Dia+16; LRV16; CSV17; Dia+17]. When the training set for a given label has been corrupted, the set of training examples for this label consists of two sub-populations. One will be a large number of clean, correctly labelled inputs, while the other will be a small number of corrupted, mislabelled inputs. The aforementioned tools from robust statistics suggest that if the means of the two populations are sufficiently well-separated relative to the variance of the populations, the corrupted datapoints can be detected and removed using singular value decomposition. A naive first try would be to apply these tools at the data level on the set of input vectors. However, as demonstrated in Figure 4-1, the high variance in the dataset means that the populations do not separate enough for these methods to work.

On the other hand, as we demonstrate in Figure 4-1, when the data points are mapped to the learned representations of the network, such a separation *does* occur. Intuitively, any feature representations for a classifier would be incentivized to boost the signal from a backdoor, since the backdoor alone is a strong indicator for classification. As the signal gets boosted, the poisoned inputs become more and more distinguished from the clean inputs. As a result, by running these robust statistics tools on the learned representation, one can detect and remove backdoored inputs. In Section 4.4, we validate these claims empirically. We demonstrate the existence of spectral signatures for backdoor attacks on image classification tasks and show that they can be used to effectively clean the corrupted training set.

Interestingly, we note that the separation requires using these recent techniques from robust statistics to detect it, even at the learned representation level. In particular, one could consider computing weaker statistics, such as ℓ_2 norms of the representations or correlations with a random vector, in an attempt to separate the clean and poisoned sub-populations. However, as shown in Figure 4-1, these methods appear to be

insufficient. While there is some separation using ℓ_2 norms, there is still substantial overlap between the norms of the learned representations of the true images and the backdoored images. The stronger guarantees from robust statistics, detailed in Section 4.3, are really necessary for detecting the poisoned inputs.

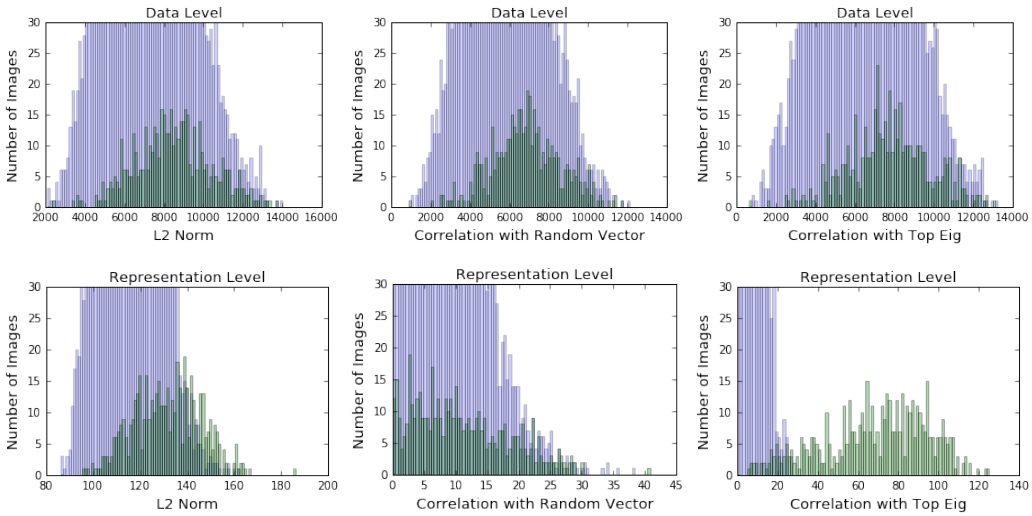


Figure 4-1: Plot of correlations for 5000 training examples correctly labelled and 500 poisoned examples incorrectly labelled. The values for the clean inputs are in blue, and those for the poisoned inputs are in green. We include plots for the computed ℓ_2 norms, correlation with a random vector, and correlation with the top singular vector of the covariance matrix of examples (respectively, representations).

4.1.2 Feature Injection

Additionally, we also propose a novel poisoning threat model. Instead of synthesizing a small number of significantly corrupted training inputs, our goal is to produce a new training dataset that differs from the original only imperceptibly, yet leads to models with significantly degraded test accuracy.

We demonstrate that deep neural networks are highly vulnerable under this threat model. However, to exploit this vulnerability we need to change our perspective on how to reduce generalization. Rather than making the training data hard to fit, we make it easily separable with a simple but poorly generalizing function.

Concretely, we formally define a new threat model for data poisoning (Section 4.5).

This model allows an adversary to perturb *all* samples in the training set but *only slightly* (rather than allowing free perturbation of a small fraction of samples).

We instantiate our attack, which we refer to as *feature injection*, on the CIFAR-10 dataset [Kri09] (Section 4.6). First, we showcase the idea in its simplest form: by overlaying imperceptible, randomly-translated, class-consistent Gaussian patterns onto training set images, we degrade the test accuracy of resulting models to below 20%. Next, we imperceptibly superimpose different digits from the MNIST dataset [LeC98] onto images from the CIFAR-10 training set, again maintaining label consistency. The resulting models are only 25% accurate on the CIFAR test set. Conversely, the models achieve almost perfect MNIST test accuracy, indicating that they do in fact rely on the injected feature. Then, we present a number of natural baselines for defending against feature injection, such as data sanitization and adversarial training; it turns out that none of these defenses are effective (Section 4.7. However, even though adversarial training itself cannot nullify injected features, the interpretable saliency maps arising from robust models (Section 3.5) can provide a means of defense. Finally, we provide a number of realistic consequences of feature injection in Section 4.8.

4.2 Finding signatures in backdoors

In this section, we describe our threat model and present our detection algorithm.

4.2.1 Threat Model

We will consider a threat model related to the work of [GDG17] in which a backdoor is inserted into the model. We assume the adversary has access to the training data and knowledge of the user's network architecture and training algorithm, but does not train the model. Rather, the user trains the classifier, but on the possibly corrupted data received from an outside source.

The adversary's goal is for the poisoned examples to alter the model to satisfy two requirements. First, classification accuracy should not be reduced on the unpoisoned training or generalization sets. Second, corrupted inputs, defined to be an attacker-

chosen perturbation of clean inputs, should be classified as belonging to a target class chosen by the adversary.

Essentially, the adversary injects poisoned data in such a way that the model predicts the true label for true inputs while also predicting the poisoned label for corrupted inputs. As a result, the poisoning is in some sense "hidden" due to the fact that the model only acts differently in the presence of the backdoor. We provide an example of such an attack in Figure 4-2. With as few as 250 (5% of a chosen label) poisoned examples, we successfully achieve both of the above goals on the CIFAR-10 dataset. Our trained models achieve an accuracy of approximately $92 - 93\%$ on the original test set, which is what a model with a clean dataset achieves. At the same time, the models classify close to 90% of the backdoored test set as belonging to the poisoned label. Further details can be found in Section 4.4. Additional examples can be found in [GDG17].

Figure 4-2: Examples of test images on which the model evaluates incorrectly with the presence of a backdoor. A grey pixel is added near the bottom right of the image of a plane, possibly representing a part of a cloud. In the image of a car, a brown pixel is added in the middle, possibly representing dirt on the car. Note that in both cases, the backdoor (pixel) is not easy to detect with the human eye. The images were generated from the CIFAR10 dataset.

4.2.2 Detection and Removal of Watermarks

We will now describe our detection algorithm. An outline of the algorithm can be found in Figure 4-3. We take a black-box neural network with some designated learned representation. This can typically be the representation from an autoencoder or a layer in a deep network that is believed to represent high level features. Then, we

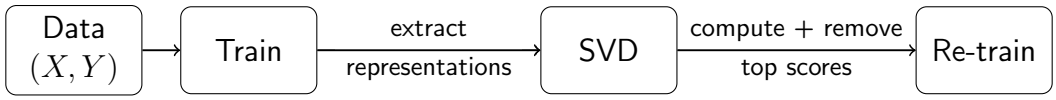


Figure 4-3: Illustration of the pipeline. We first train a neural network on the data. Then, for each class, we extract a learned representation for each input from that class. We next take the singular value decomposition of the covariance matix of these representations and use this to compute an outlier score for each example. Finally, we remove inputs with the top scores and re-train.

take the representation vectors for all inputs of each label. The intuition here is that if the set of inputs with a given label consists of both clean examples as well as corrupted examples from a different label set, the backdoor from the latter set will provide a strong signal in this representation for classification. As long as the signal is large in magnitude, we can detect it via singular value decomposition and remove the images that provide the signal. In Section 4.3, we formalize what we mean by large in magnitude.

More detailed pseudocode is provided in Algorithm 1.

Algorithm 1

Input: Training set $\mathcal{D}_{\text{train}}$, randomly initialized neural network model $\mathcal{L}$ providing a feature representation $\mathcal{R}$, and upper bound on number of poisoned training set examples ε. For each label y of $\mathcal{D}_{\text{train}}$, let $\mathcal{D}_y$ be the training examples corresponding to that label.
Train $\mathcal{L}$ on $\mathcal{D}_{\text{train}}$.
Initialize $S \leftarrow \{\}$.
for all y **do**
  Set $n = |\mathcal{D}_y|$, and enumerate the examples of $\mathcal{D}_y$ as $x_1, \ldots, x_n$.
  Let $\widehat{\mathcal{R}} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}(x_i)$.
  Let $M = [\mathcal{R}(x_i) - \widehat{\mathcal{R}}]_{i=1}^{n}$ be the $n \times d$ matrix of centered representations.
  Let v be the top right singular vector of M.
  Compute the vector τ of *outlier scores* defined via $\tau_i = \left((\mathcal{R}(x_i) - \widehat{\mathcal{R}}) \cdot v\right)^2$.
  Remove the examples with the top $1.5 \cdot \varepsilon$ scores from $\mathcal{D}_y$.
  $S \leftarrow S \cup \mathcal{D}_y$
end for
$\mathcal{D}_{\text{train}} \leftarrow S$.
Re-train $\mathcal{L}$ on $\mathcal{D}_{\text{train}}$ from a random initialization.
Return $\mathcal{L}$.

4.3 Spectral signatures for backdoored data in learned representations

In this section we give more rigorous intuition as to why learned representations on the corrupted data may cause the attack to have a detectable spectral signature.

4.3.1 Outlier removal via SVD

We first give a simple condition under which spectral techniques are able to reliably detect outliers:

Definition 4.3.1. Fix $1/2 > \varepsilon > 0$. Let D, W be two distributions with finite covariance, and let $F = (1-\varepsilon)D + \varepsilon W$ be the mixture of D, W with mixing weights $(1-\varepsilon)$ and ε, respectively. We say that D, W are *ε-spectrally separable* if there exists a $t > 0$ so that

$$\Pr_{X \sim D}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| > t\right] < \varepsilon$$
$$\Pr_{X \sim W}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| < t\right] < \varepsilon \ ,$$

where v is the top eigenvector of the covariance of F.

Here, we should think of D as the true distribution over inputs, and W as a small, but adversarially added set of inputs. Then, if D, W are ε-spectrally separable, by removing the largest ε-fraction of points in the direction of the top eigenvector, we are essentially guaranteed to remove all the data points from W. Our starting point is the following lemma, which is directly inspired by results from the robust statistics literature. While these techniques are more or less implicit in the robust statistics literature, we include them here to provide some intuition as to why spectral techniques should detect deviations in the mean caused by a small sub-population of poisoned inputs.

Lemma 1. *Fix $1/2 > \varepsilon > 0$. Let D, W be distributions with mean $\boldsymbol{\mu}_D, \boldsymbol{\mu}_W$ and covariances $\boldsymbol{\Sigma}_D, \boldsymbol{\Sigma}_W \preceq \sigma^2 I$, and let $F = (1-\varepsilon)D + \varepsilon W$. Then, if $\|\boldsymbol{\mu}_D - \boldsymbol{\mu}_W\|_2^2 \geq \frac{6\sigma^2}{\varepsilon}$, then D, W are ε-spectrally separable.*

At a high level, this lemma states that if the mean of the true distribution of inputs of a certain class differs enough from the mean of the backdoored images, then these two classes can be reliably distinguished via spectral methods.

We note here that Lemma 1 is stated for population level statistics; however, it is quite simple to convert these guarantees to standard finite-sample settings, with optimal sample complexity. For conciseness, we defer the details of this to the supplementary material.

Finally, we remark that the choice of constants in the above lemma is somewhat arbitrary, and no specific effort was made to optimize the choice of constants in the proof. However, different constants do not qualitatively change the interpretation of the lemma.

The rest of this section is dedicated to a proof sketch of Lemma 1. The omitted details can be found the supplementary material.

Proof sketch of Lemma 1. By Chebyshev's inequality, we know that

$$\Pr_{X \sim D}[|\langle X - \boldsymbol{\mu}_D, u\rangle| > t] \leq \frac{\sigma^2}{t^2}\ , \text{ and} \tag{4.1}$$

$$\Pr_{X \sim W}[|\langle X - \boldsymbol{\mu}_W, u\rangle| > t] \leq \frac{\sigma^2}{t^2}\ , \tag{4.2}$$

for any unit vector u. Let $\Delta = \boldsymbol{\mu}_D - \boldsymbol{\mu}_W$, and recall v is the top eigenvector of $\boldsymbol{\Sigma}_F$. The "ideal" choice of u in (4.1) and (4.2) that would maximally separate points from D and points from W would be simply a scaled version of Δ. However, one can show that any unit vector which is sufficiently correlated to Δ also suffices:

Lemma 2. *Let $\alpha > 0$, and let u be a unit vector so that $|\langle u, \Delta\rangle| > \alpha \cdot \sigma/\sqrt{\varepsilon}$. Then there exists a $t > 0$ so that*

$$\Pr_{X \sim D}[|\langle X - \boldsymbol{\mu}_D, u\rangle| > t] < \varepsilon\ , \textit{ and } \Pr_{X \sim W}[|\langle X - \boldsymbol{\mu}_W, u\rangle| > t] < \frac{1}{(\alpha-1)^2}\varepsilon\ .$$

The proof of this is deferred to the supplementary material.

What remains to be shown is that v satisfies this condition. Intuitively, this works because Δ is sufficiently large that its signal is noticeable in the spectrum of $\mathbf{\Sigma}_F$. As a result, v (being the top eigenvector of $\mathbf{\Sigma}_F$) must have non-negligible correlation with Δ. Concretely, this allows us to show the following lemma, whose proof we defer to the supplementary material.

Lemma 3. *Under the assumptions of Lemma 1, we have* $\langle v, \Delta \rangle^2 \geq \frac{2\sigma^2}{\varepsilon}$.

Finally, combining Lemmas 2 and 3 imply Lemma 1. □

4.4 Experiments

4.4.1 Setup

We study backdoor poisoning attacks on the CIFAR10 [Kri09] dataset, using a standard ResNet [He+16] model with 3 groups of residual layers with filter sizes $[16, 16, 32, 64]$ and 5 residual units per layer. Unlike more complicated feature extractors such as autoencoders, the standard ResNet does not have a layer tuned to be a learned representation for any desired task. However, one can think of any of the layers as modeling different kinds of representations. For example, the first convolutional layer is typically believed to represent edges in the image while the latter layers learn "high level" features [Don+14]. In particular, it is common to treat the last few layers as representations for classification.

Our experiments showed that our outlier removal method successfully removes the backdoor when applied on many of the later layers. We choose to report the results for the second to last residual unit simply because, on average, the method applied to this layer removed the most poisoned images. We also remark that we tried our method directly on the input. Even when data augmentation is removed, so that the backdoor is not flipped or translated, the signal is still not strong enough to be detected, suggesting that a learned representation amplifying the signal is really necessary.

We note that we also performed the outlier removal on a VGG [SZ15] model.

4.4.2 Attacks

Our standard attack setup consists of a pair of (attack, target) labels, a backdoor shape (pixel, X, or L), an epsilon (number of poisoned images), a position in the image, and a color for the mark.

For our experiments, we choose 4 pairs of labels by hand- (airplane, bird), (automobile, cat), (cat, dog), (horse, deer)- and 4 pairs randomly- (automobile, dog), (ship, frog), (truck, bird), (cat,horse). Then, for each pair of labels, we generate a random shape, position, and color for the backdoor. We also use the hand-chosen backdoors of Figure 4-2.

4.4.3 Attack Statistics

In this section, we show some statistics from the attacks that give motivation for why our method works. First, in the bottom right plot of Figure 4-1, we can see a clear separation between the scores of the poisoned images and those of the clean images. This is reflected in the statistics displayed in Table 4.1. Here, we record the norms of the mean of the representation vectors for both the clean inputs as well as the clean plus corrupted inputs. Then, we record the norm of the difference in mean to measure the shift created by adding the poisoned examples. Similarly, we have the top three singular values for the mean-shifted matrix of representation vectors of both the clean examples and the clean plus corrupted examples. We can see from the table that there is quite a significant increase in the singular values upon addition of the poisoned examples. The statistics gathered suggest that our outlier detection algorithm should succeed in removing the poisoned inputs.

4.4.4 Evaluating our Method

In Tables 4.2, we record the results for a selection of our training iterations. For each experiment, we record the accuracy on the natural evaluation set (all 10000 test

Experiment	Norm of Mean	Shift in Mean	1st SV	2nd SV	3rd SV
Birds only	78.751	N/A	1194.223	1115.931	967.933
Birds + planes	78.855	6.194	1613.486	1206.853	1129.711
Cats + cars	89.409	N/A	1016.919	891.619	877.743
Cats + poison	89.690	7.343	1883.934	1030.638	913.895

Table 4.1: We record statistics for the two experiments coming from Figure 4-2, backdoored planes labelled as birds and backdoored cars labelled as cats. For both the clean dataset and the clean plus poisoned dataset, we record the norm of the mean of the representation vectors and the top three singular values of the covariance matrix formed by these vectors. We also record the norm of the difference in the means of the vectors from the two datasets.

images for CIFAR10) as well as the poisoned evaluation set (1000 images of the attack label with a backdoor). We then record the number of poisoned images left after one removal step and the accuracies upon retraining. The table shows that for a variety of parameter choices, the method successfully removes the attack. Specifically, the clean and poisoned test accuracies for the second training iteration after the removal step are comparable to those achieved by a standard trained network on a clean dataset. For reference, a standard trained network on a clean training set classifies a clean test set with accuracy 92.67% and classifies each poisoned test set with accuracy given in the rightmost column of Table 4.2.

We also reran the experiments multiple times with different random choices for the attacks. For each run that successfully captured the backdoor in the first iteration, which we define as recording approximately 90% or higher accuracy on the poisoned set, the results were similar to those recorded in the table. As an aside, we note that 5% poisoned images is not enough to capture the backdoor according to our definition in our examples from Figure 4-2, but 10% is sufficient.

4.4.5 Sub-populations

Our outlier detection method crucially relies on the difference in representation between the clean and poisoned examples being much larger than the difference in representations within the clean examples. An interesting question to pose, then, is

Sample	Target	Epsilon	Nat 1	Pois 1	# Pois Left	Nat 2	Pois 2	Std Pois
	bird	5%	92.27%	74.20%	57	92.64%	2.00%	1.20%
		10%	92.32%	89.80%	7	92.68%	1.50%	
	bird	5%	92.49%	98.50%	0	92.76%	2.00%	1.90%
		10%	92.55%	99.10%	0	92.89%	0.60%	
	bird	5%	92.66%	89.50%	14	92.59%	1.40%	1.10%
		10%	92.63%	95.50%	2	92.77%	0.90%	
	cat	5%	92.45%	83.30%	24	92.24%	0.20%	0.10%
		10%	92.39%	92.00%	0	92.44%	0.00%	
	cat	5%	92.60%	95.10%	1	92.51%	0.10%	0.10%
		10%	92.83%	97.70%	1	92.42%	0.00%	
	cat	5%	92.80%	96.50%	0	92.77%	0.10%	0.00%
		10%	92.74%	99.70%	0	92.71%	0.00%	
	dog	5%	92.91%	98.70%	0	92.59%	0.00%	0.00%
		10%	92.51%	99.30%	0	92.66%	0.10%	
	dog	5%	92.17%	89.80%	7	93.01%	0.00%	0.00%
		10%	92.55%	94.30%	1	92.64%	0.00%	
	horse	5%	92.38%	96.60%	0	92.87%	0.80%	0.80%
		10%	92.72%	99.40%	0	93.02%	0.40%	
	horse	5%	92.60%	99.80%	0	92.57%	1.00%	0.80%
		10%	92.26%	99.80%	0	92.63%	1.20%	
	cat	5%	92.68%	97.60%	1	92.72%	8.20%	7.20%
		10%	92.59%	99.00%	4	92.80%	7.10%	
	cat	5%	92.86%	98.60%	0	92.79%	8.30%	8.00%
		10%	92.29%	99.10%	0	92.57%	8.20%	
	deer	5%	92.68%	99.30%	0	92.68%	1.10%	1.00%
		10%	92.68%	99.90%	0	92.74%	1.60%	
	deer	5%	93.25%	97.00%	1	92.75%	2.60%	1.10%
		10%	92.31%	97.60%	1	93.03%	1.60%	
	frog	5%	92.87%	88.80%	10	92.61%	0.10%	0.30%
		10%	92.82%	93.70%	3	92.74%	0.10%	
	frog	5%	92.79%	99.60%	0	92.71%	0.20%	0.20%
		10%	92.49%	99.90%	0	92.58%	0.00%	
	bird	5%	92.52%	97.90%	0	92.69%	0.00%	0.00%
		10%	92.68%	99.30%	0	92.45%	0.50%	
	bird	5%	92.51%	87.80%	1	92.66%	0.20%	0.00%
		10%	92.74%	94.40%	0	92.91%	0.10%	

Table 4.2: Full table of accuracy and number of poisoned images left for different attack parameters. For each attack to target label pair, we provide a few experimental runs with different backdoor.

what happens when the variance in representations within clean examples increases. A natural way this may happen is by combining labels; for instance, by combining "cats" and "dogs" into a shared class called "pets". When this happens, the variance in the representations for images in this shared class increases. How robust are our methods to this sort of perturbation? Do spectral signatures arise even when the variance in representations has been artificially increased?

In this section, we provide our experiments exploring our outlier detection method when one class class consists of a heterogenous mix of different populations. As mentioned above, we combined "cats" and "dogs" into a class we call "pets". Then, we install a backdoor of poisoned automobiles labeled as pets, as well as poisoned pets labeled as automobiles. With these parameters, we train our Resnet and perform outlier detection. The results are provided in Table 4.3. We can see from these results that in both cases, the automobile examples still have a representation sufficiently separated from the combined cats and dogs representations.

4.5 Approach and Threat Model

Leveraging the aforementioned observations, we propose a class of attacks called *feature injection attacks*. In all these attacks, the adversary injects class-consistent features into every input in the dataset, with the goal of making that dataset incredibly *easy* to fit. This goal is in stark contrast to classical data poisoning, in which the adversary typically adds a small number of points that are *hard* to fit. Leveraging the fact that deep neural networks tend to learn simpler functions first [VCL19; Jac+19], we encourage the model to fit the entire poisoned dataset with a simple, yet poorly-generalizing function that relies on the injected features. Then, at test time, the trained model fails to correctly predict labels of unseen inputs, as these inputs lack the easy-to-learn, class-consistent features injected into the training set.

We provide an illustration of a feature injection attack in Figure 4-4. In the original dataset, the blue and red data points are interwoven within one another, making a (well-generalizing) separator feasible but difficult to find. The adversary then modifies

Sample	Target	Epsilon	Nat 1	Pois 1	# Pois Left	Nat 2	Pois 2
	pets	5%	93.99%	95.80%	0	94.18%	0.30%
		10%	94.05%	96.70%	0	94.27%	0.00%
	pets	5%	94.28%	95.00%	0	94.12%	0.20%
		10%	94.13%	99.70%	0	93.89%	0.00%
	pets	5%	94.12%	89.80%	0	94.18%	0.10%
		10%	93.90%	93.40%	0	94.11%	0.10%
	pets	5%	93.97%	94.80%	0	94.42%	0.00%
		10%	94.23%	97.20%	0	93.96%	0.30%
	automobile	5%	93.96%	98.65%	0	94.46%	0.20%
		10%	94.18%	99.20%	0	94.00%	0.20%
	automobile	5%	94.20%	99.15%	0	94.36%	0.25%
		10%	94.03%	99.55%	0	94.03%	0.10%
	automobile	5%	93.89%	94.40%	6	94.20%	0.20%
		10%	94.49%	97.20%	2	94.49%	0.05%
	automobile	5%	94.26%	95.60%	5	94.06%	0.00%
		10%	94.20%	98.45%	1	94.06%	0.15%

Table 4.3: Results for a selection of different attack parameters on a combined label of cats and dogs, that we call pets. Natural and poisoned accuracy are reported for two iterations, before and after the removal step. The attack parameters are given by a backdoor attack image, target label, and percentage of added images.

this dataset by injecting a new predictive dimension into the dataset, with a different value for blue and red points (in the original dataset, this dimension was constant for all inputs), the two classes are separable with a simple hyperplane bearing no resemblance to the original optimal classifier (in fact, this hyperplane achieves an accuracy no better than random on natural data).

Implementing our proposed data poisoning strategy requires rethinking the data poisoning threat model. Rather than letting an adversary arbitrarily corrupt a small fraction of samples (as in traditional data poisoning setting), in our proposed threat model the adversary can modify every input, but only slightly[1]. That is, given a training set $D = \{(x_i, y_i)\}$, an adversary outputs a new training set $D_{adv} = \{(x'_i, y_i)\}$, such that

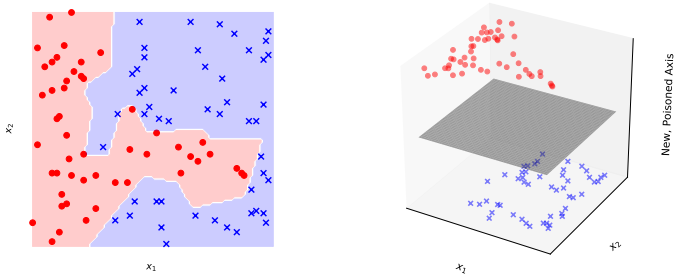
$$d(x'_i, x_i) < \varepsilon,$$

for a pre-specified distance function $d(\cdot, \cdot)$ and perturbation budget ε. Note that (for reasonable choice of d and ε), attacks under this threat model are virtually undetectable by human observers (in a similar manner to adversarial examples). Throughout this paper, we instantiate our new threat model to an ℓ_2–based notion of imperceptibility; that is, we choose $d(x, y) = \|x - y\|_2$ to be Euclidean distance, and choose ε such that the perturbed training set looks completely natural to a human observer.

4.6 Attack Implementations

In the previous section, we outlined a class of attacks based on *feature injection*, and a corresponding threat model under which these attacks can be carried out without human detection. Here, we instantiate two simple but illustrative examples of feature injection attacks on the CIFAR-10 dataset. For each CIFAR-10 label, we overlay 'easy-to-learn" and consistent patterns on all inputs corresponding to that label. At training time, models quickly learn these meaningless correlations between patterns and labels, and ultimately correctly classify the training set without actually learning

[1]We note that for very simple families of distributions (much simpler than those induced by image data), this definition can actually be reduced to the ε-corruption model via a coupling argument.



(a) Original Dataset (b) Corrupted Dataset

Figure 4-4: A toy model in which a dataset is poisoned with elementary, class consistent features. The original dataset consists of an entangled mixture of blue and red points with a complicated separator. These points are three dimensional, but all points lie in a two dimensional slice. In the poisoned dataset, the adversary injects all the points with a distinct, class consistent value in the third, previously constant dimension. After the poisoning, a simple hyperplane suffices as a separator. This hyperplane is not a generalizing classifier, however, as it fails to correctly classify inputs from the original distribution, in which the third coordinate is not class predictive.

to use any salient features of natural CIFAR images. Thus, when tested on unseen natural data, the models cannot consistently predict the correct CIFAR labels.

Feature injection via Gaussian patterns. To showcase the simplicity of feature injection, we introduce Gaussian patterns into the CIFAR10 training set. For each class label, we sample an isotropic multivariate Gaussian of the same shape as a CIFAR-10 image. We then normalize the Gaussian to a fixed magnitude, and apply a random translation of magnitude T in a random direction (where T is a hyperparameter). The translated normalized Gaussian is then superimposed on all training images of the appropriate label. We will be more interested in the case where $T > 0$, as when $T = 0$ the dataset is actually linearly separable (and thus is detectable by training a linear classifier on the training set). Throughout our preliminary experimentation, we use standard training procedures for each task; for example, we use typical data augmentation methods.

Table 4.4 shows that standard training is vulnerable to even this simple instantiation of feature injection. We vary the norms of the added Gaussians (ε) and whether to

allow translations. In all settings, the network learns the Gaussian patterns and is unable to generalize to the test set.

Norm	Fixed	Shift
1.0	18.73%	13.34%
3.0	12.08%	17.58%
5.0	9.73%	17.82%

Table 4.4: Accuracy on test set for models trained with added gaussians of varying norm and either with a shift or without one. Test accuracy is reported after training has converged. All models train with standard data augmentation. None of the models generalize to the test set.

Feature injection via MNIST digits. In this section, we demonstrate that more complicated features than Gaussians can still successfully degrade classification accuracy. First, we form a bijection between CIFAR-10 classes and MNIST digits. Then, for each training input from CIFAR-10, we randomly and independently sample an MNIST input with the corresponding label. Note that, unlike the previously presented Gaussian feature injection, MNIST feature injection allows for different images in the same CIFAR-10 class to be overlaid with different MNIST inputs.

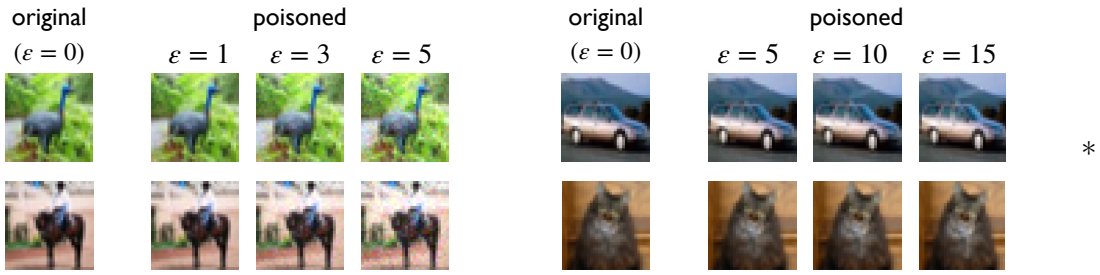
As indicated in Table 4.5 the resulting models consistently fails to generalize to the CIFAR test set. However, the model successfully generalizes to the MNIST test set. As in the case of the Gaussian addition, the norms of the added digits are varied, maintaining semantic integrity of the original CIFAR input, and a potential translational shift of up to one pixel is allowed.

The success of this method appears to be a byproduct of MNIST being significantly easier to fit than CIFAR-10. Rather than learning the actual CIFAR data, the model quickly learns to classify the overlayed MNIST dataset. Once perfect training accuracy is achieved from using the easier-to-learn MNIST features, the model stops receiving meaningful gradient updates. The more complex features from CIFAR are never taken into consideration.

	CIFAR Test		MNIST Test	
Norm	**Fixed**	**Shift**	**Fixed**	**Shift**
5.0	34.81%	32.31%	97.09%	96.10%
10.0	20.67%	24.10%	99.14%	99.13%
15.0	18.95%	25.55%	98.23%	96.98%

Table 4.5: Accuracy on CIFAR and MNIST test sets for models trained on CIFAR with added MNIST images of varying norm. The test accuracy is reported after training has converged. All models are trained with standard data augmentation.

We remark that the added norms for the Gaussian patterns vastly differ from those for the MNIST digits. The reason for this difference is that a larger magnitude MNIST digit can be added to CIFAR images without visibly altering the original input, relative to added Gaussians. MNIST digits typically consist of a background of mostly black, i.e. 0s, and a white foreground. This serves to only whiten small portions of the CIFAR images and does not cause the new data to look obviously suspicious. On the other hand, a Gaussian is varied and multi-colored, and so a large-norm Gaussian cannot be added without perceptibly modifying the CIFAR inputs. We provide examples of the resulting images from these additions in Figure 4-5 for comparison.



(a) Gaussian Patterns (b) MNIST Digits

Figure 4-5: Examples of the resulting images from adding patterns of varying norms onto CIFAR inputs. On the left, we add class-consistent, randomly generated Gaussians, and on the right, MNIST digits of the corresponding label. The norms we allow are different in the two methods because, as can be seen from the figure, MNIST digits of much higher norm can be added without meaningfully changing the CIFAR input.

Feature injection on a subset of classes. For completeness, we also experiment with injecting features onto images from only a subset of classes. At first glance, one may expect that feature injection might fail in this setting, as the model is required to learn the "right" features for the non-poisoned classes, which may in turn help the model generalize at least partially to the poisoned labels. (For instance, if the "car" class has been poisoned, the model may still learn the "wheel" feature from the truck class and then use it to improve classification of cars.) However, our experimental results, provided in Table 4.6, indicate that this is not the case; it is possible for an adversary to reduce model accuracy on specific classes using feature injection. As we discuss in Section 4.8, these partial feature injection attacks are of practical importance, as they enable data poisoning attacks in settings where data is collected from many sources.

4.6.1 Gaussian Addition on a Subset of Classes

		Poisoned	Clean
# Clean Labels	1	0.00%	100.00%
	3	0.16%	92.40%
	5	0.00%	88.94%
	7	0.00%	85.60%
	9	0.00%	85.91%

Table 4.6: Accuracy on test set for poisoned and clean (un-poisoned) classes, where the poisoned classes have a Gaussian of norm 1 added to them. All Gaussians are allowed a shift of one in any translational direction.

4.7 Baseline Defenses

In this section, we present a number of baseline defenses for defending against feature injection. Despite the simplicity of our attacks, our results demonstrate that they are surprisingly hard to mitigate, even with full knowledge of the attack algorithm. While

these results do not imply that no effective defense exists, they suggest that a more involved and complex training procedure is required to protect against such poisoning attacks. For each defense idea, we choose to focus on the added Gaussians, as they exhibit feature injection in its simplest form.

First, in Section 4.7.1, we implement elementary data sanitization procedures: PCA on the dataset, training with a linear classifier, and saliency maps. Then, in Section 4.7.2, we adversarially train on the poisoned datasets to evaluate whether robust models learn to ignore the injected features.

4.7.1 Data Sanitization

As data poisoning from any threat model deals with corrupting the dataset, a natural defense is to simply clean the data with various data sanitization techniques. We bring into consideration three such protocols here and demonstrate that feature injection bypasses all of them. For each such technique, we apply the same training procedure. First, we apply the protocol to compute a direction in input space for each class. Then, we project every input of the training data away from these directions. If these directions have high correlation with the Gaussians, then the model learns to classify without the Gaussians. In other words, the model would behave as normal, classifier.

We detail the data sanitization protocols in the following:

Data Sanitization

- **PCA-Principal Component Analysis.** In the Gaussian addition model, the same pattern is added to every input from a corresponding CIFAR class. Thus, one might expect that PCA, a method for finding important directions in a dataset would retrieve these patterns.

- **Linear Classification.** We initially train a linear classifier on the poisoned data. Since every class contains a different Gaussian, if the linear model is successful, the weights of the model will correspond to the Gaussians.

- **Saliency Maps.** For this method, we perform one training iteration with our primary model architecture. Here, we suspect that the poisoning has succeeded,

which means our model has learned to classify based on the Gaussian patterns. As a result, if we compute the saliency maps per input, that is, the direction in input space that most drastically affects loss, we may be able to recover the patterns.

In Figure 4-6, we display the correlation of the data sanitization protocols' computed directions onto the Gaussian features. For reference, we also include a baseline, which is the average correlation between two randomly chosen directions in input space. From the figure, we can see that **linear classification** finds directions that have higher than random correlation with the added noise. Also, as one would intuitively predict, the correlation is higher for the fixed case than when the patterns are allowed a translational shift. Then, in Table 4.7, we provide the resulting test accuracy from training after projecting the images away from the computed directions. Even when the Gaussian has norm 1 and is not allowed a translational shift, **PCA** and **saliency maps** are unable to find directions with high correlation onto the patterns. That is, they are unable to recover high test accuracy. On the other hand, **linear classification** is able to recover some accuracy when the Gaussians have norm 1, but unsurprisingly, only when there is no translational shift. Even though the weights of the linear model have high correlation onto the added Gaussians, a large enough residue pattern remains after projection when the initial Gaussians have higher norm. In these instances, projection away from the linear classification weights does not allow for the model to recover high test accuracy.

Figure 4-6: Visualization of the correlations between the Gaussian noise directions and the directions computed from the various data sanitization methods.

Norm	Fixed			Shift		
	PCA	**Linear**	**Saliency**	**PCA**	**Linear**	**Saliency**
1.0	16.41%	84.73%	17.18%	49.96%	81.77%	10.42%
3.0	12.33%	14.53%	13.50%	18.67%	10.50%	11.31%
5.0	9.97%	14.80%	13.61%	16.26%	9.97%	11.39%

Table 4.7: Accuracy on the test set for models trained on poisoned data after projection away from directions computed with the data sanitization protocols.

4.7.2 Adverarial Training

Another perspective one can take on feature injection is to view the patterns as perturbations of the input. By taking this perspective, the proposed data poisoning threat model can be fit into the realm of adversarial training. Adversarial training is designed to encourage networks to be invariant to small perturbations of the input- for some chosen norm- and to smooth out any specific directions that heavily influence classification. As such, these models could feasibly learn to ignore any Gaussian patterns, as long as they are not too large, and instead learn the original input's features. We choose to use models robust to ℓ_2 perturbations, as the perturbations added to poison training images is restricted in ℓ_2 norm.

Norm	Fixed		Shift	
	$\boldsymbol{\varepsilon = 1.0}$	$\boldsymbol{\varepsilon = 2.0}$	$\boldsymbol{\varepsilon = 1.0}$	$\boldsymbol{\varepsilon = 2.0}$
1.0	84.99%	69.57%	82.76%	70.06%
3.0	12.74%	32.07%	31.23%	72.54%
5.0	13.72%	10.38%	11.93%	24.34%

Table 4.8: Accuracy on the test set for models trained with an L2 adversary of varying ε, against added Gaussians of varying norm. We note that even though the Gaussians can have norm 5, a model cannot be adversarially trained with $\varepsilon = 5$.

Our experimental results indicate that adversarial training can somewhat mitigate the effects of feature injection, but not completely. Specifically, we see that when the

set of perturbations, dictated by the value of ε, contains the Gaussian patterns, high test accuracy is recovered. However, the max norm of an added Gaussian pattern that maintains semantic integrity of the inputs yields a perturbation outside the scope allowed for adversarial training. For instance, in the case of ℓ_2 perturbations, we have demonstrated that adding a pattern of norm 5 does not visibly alter the training images in any meaninful way. On the other hand, there exists a perturbation of norm 5 that can semantically change the class of a training input. Attempting to train a robust model with $\varepsilon = 5$ yields a constant classifier, as the task is infeasible.

Even though, adversarial training cannot completely detect injected features, one promising direction is to utilize their interpretable saliency maps (Section 3.5). In particular, for each model of Table 4.8, we compute the saliency maps of each input and measure the correlation onto the injected features. As we see in Table 4.9, the saliency correlations of robustly trained models are significantly hire that those of naturally trained networks. It is still open, however, to utilize these saliency maps in forming a poisoning defense.

Norm	**Fixed**			**Shift**		
	$\boldsymbol{\varepsilon = 0}$	$\boldsymbol{\varepsilon = 1.0}$	$\boldsymbol{\varepsilon = 2.0}$	$\boldsymbol{\varepsilon = 0}$	$\boldsymbol{\varepsilon = 1.0}$	$\boldsymbol{\varepsilon = 2.0}$
1.0	0.0581	0.3609	0.0583	0.0268	0.0574	0.1105
3.0	0.0557	0.1537	0.3465	0.0218	0.0788	0.2303
5.0	0.0748	0.1435	0.2314	0.0178	0.0549	0.1105

Table 4.9: Correlations of saliency maps computed from trained models with injected features. Results are shown for both standard ($\varepsilon = 0$) and robust models.

4.8 Applications

In this chapter, we proposed a new method for poisoning a dataset. This method prevents classifiers trained on the dataset from learning salient, well-generalizing features. Here we outline potential applications of this method which can be either malicious or benign in nature.

Accuracy degradation attacks. Standard deep learning methods require a large amount of data to achieve state-of-the-art performance. Since large high-quality datasets can be hard to obtain, practitioners often rely on data from untrusted sources. Hence an adversary could access the training set and add imperceptible perturbations to the inputs. Such perturbations would ruin the dataset for the purpose of accurately predicting new, unseen samples.

Hiding training data. In certain cases, actors may be required (i.e. a company following government regulations) to provide the data used for training a model. However, since that data could be a competitive advantage, it may be desirable to prevent other entities from training on this data. Our method provides a way to release data that appears perceptually identical to the original, but is not useful for learning useful classifiers.

Face Recognition Systems. When people release photos of themselves, they run the risk of these photos being used to train machine learning systems involuntarily. One can use our method to plant a simple, yet spurious association between their photos and their identity. This allows one to evade detection by simply not introducing the pattern in the new photos.

4.9 Related Work

Data poisoning for classical models. There has been extensive work in modeling threats to machine learning systems that involve adversary access to training data. The classical data poisoning formulation has its roots in robust estimation under the Huber contamination model. The goal of a robust algorithm in this setting is to estimate a parameter θ of a distribution P_θ given samples from $P_{adv} = (1-\varepsilon)\cdot P_\theta + \varepsilon \cdot Q$; i.e., in the presence of an adversary that randomly inserts samples from Q at a rate of ε. Previous work in data poisoning has also considered the stronger ε-corruption model, under which the adversary is allowed to corrupt an ε-fraction of samples drawn from P_θ, but is now left to decide *which points* to corrupt and how. Through a simple

coupling argument, one can show that this corruption model actually captures all $\|P_\theta - P_{adv}\|_{TV} \leq \varepsilon$.

A rich body of work has demonstrated the vulnerability of classical machine learning models to ε-corruption attacks. Generalization performance on these models (e.g. mean estimation, linear classification, SVM, etc.) can be drastically degraded with just a few additional or corrupted training samples [BNL12; KL17; MZ15; SKL17; Xia+15].

Backdoor Attacks To the best of our knowledge, the first instance of backdoor attacks for deep neural networks appeared in [GDG17]. The ideas for their attacks form the basis for our threat model and are also used in [Che+17b].

Another line of work on data poisoning deal with attacks that are meant to degrade the model's generalization accuracy. The idea of influence functions [KL17] provides a possible way to detect such attacks, but do not directly apply to backdoor attacks which do not cause misclassification on typical test examples. The work in [Sha+18] creates an attack that utilizes data poisoning poisoning in a different way. While similar in some ways to the poisoning we consider, their corruption attempts to degrade the model's test performance rather than install a backdoor. Outlier removal defenses are studied in [SKL17], but while our methods detect and remove outliers of a certain kind, their evaluation only applies in the test accuracy degradation regime.

Another line of work focuses on applying the robust statistics tools developed in [Dia+16; LRV16; CSV17; Dia+17] to robust stochastic optimization problems [Bal+17; CSV17; Dia+18; KKM18; Pra+18]. Again, the defenses in these papers target attacks that degrade test accuracy. Nonetheless, for completeness, we checked and found that these techniques were unable to reliably detect the corrupted data points. Independent work by [LDG18] proposes another approach to protection against backdoor attacks that relies on a certain type of neuron pruning, as well as re-training on clean data.

Learning simple patterns. Several prior works have noticed models' ability to learn simple patterns in the training data. First, label leaking [KGB17] observes

that models adversarially trained with FGSM can achieve 100% "adversarial accuracy" (against FGSM) and 0% standard accuracy by relying on memorizing the adversarial patterns introduced by FGSM themselves. In a similar vein, Jacobsen et al. [Jac+19] show this by introducing predictive shifts to the MNIST dataset and demonstrating that the resulting models become highly invariant to salient features in the input. Our proposed approach to data poisoning formulates these observations into a concrete security threat, and shows that standard ML systems and baseline defenses are highly vulnerable to the corresponding attacks.

4.10 Discussion and Conclusion

In this chapter, we present the notion of spectral signatures and demonstrate how they can be used to detect backdoor poisoning attacks. Our method relies on the idea that learned representations for classifiers amplify signals crucial to classification. Since the backdoor installed by these attacks change an example's label, the representations will then contain a strong signal for the backdoor. Based off this assumption, we then apply tools from robust statistics to the representations in order to detect and remove the poisoned data.

We implement our method for the CIFAR10 image recognition task and demonstrate that we can detect outliers on real image sets. We provide statistics showing that at the learned representation level, the poisoned inputs shift the distribution enough to be detected with SVD methods. Furthermore, we also demonstrate that the learned representation is indeed necesary; naively utilizing robust statistics tools at the data level does not provide a means with which to remove backdoored examples.

One interesting direction from our work is to further explore the relations to adversarial examples. As mentioned previously in the paper, models robust to a group of perturbations are then robust to backdoors lying in that group of perturbations. In particular, if one could train a classifier robust to ℓ_0 perturbations, then backdoors consisting of only a few pixels would not be captured.

In general, we view the development of classifiers resistant to data poisoning as

a crucial step in the progress of deep learning. As neural networks are deployed in more situations, it is important to study how robust they are, especially to simple and easy to implement attacks. This paper demonstrates that machinery from robust statistics and classical machine learning can be very useful tools for understanding this behavior. We are optimistic that similar connections may have widespread application for defending against other types of adversarial attacks in deep learning.

Additionally, we proposed a new approach to data poisoning for deep learning models. Rather than only allowing the introduction of a small number of corrupted samples, we instead restrict the size of the perturbation allowed on each clean input. The success of the approach relies on the fact that deep learning models learn features in increasing order of difficulty. Consequently, we inject easy-to-learn features into the data so that the model achieves perfect classification before learning any true features. We first gave explicit examples of this feature injection in the form of class-consistent, randomly generated Gaussian patterns. Then, we showed that more complicated patterns, specifically MNIST digits, can be added as long as they are still simpler than the clean data.

Bibliography

[Aba+16] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2016.

[ACW18] Anish Athalye, Nicholas Carlini, and David A. Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *International Conference on Machine Learning (ICML)*. 2018.

[Ade+18] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[Adi+18] Yossi Adi et al. "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring". In: *arXiv preprint arXiv:1802.04633*. 2018.

[Ath+18] Anish Athalye et al. "Synthesizing Robust Adversarial Examples". In: *International Conference on Machine Learning (ICML)*. 2018.

[Bal+17] Sivaraman Balakrishnan et al. "Computationally efficient robust sparse estimation in high dimensions". In: *Conference on Learning Theory*. 2017, pp. 169–212.

[Bau+19a] David Bau et al. "GAN Dissection: Visualizing and Understanding Generative Adversarial Networks". In: *International Conference on Learning Representations (ICLR)*. 2019.

[Bau+19b] David Bau et al. "Inverting Layers of a Large Generator". In: *ICLR Debugging Machine Learning Models Workshop*. 2019.

[BB19] Wieland Brendel and Matthias Bethge. "Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet". In: *International Conference on Learning Representations (ICLR)*. 2019.

[BCN06] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. 2006.

[BCV13] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: (2013).

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations (ICLR)*. 2019.

[Beh+18] Jens Behrmann et al. "Invertible residual networks". In: *arXiv preprint arXiv:1811.00995*. 2018.

[Ben19] Yoshua Bengio. *Talk Abstract: Learning High-Level Representations for Agents*. Abstract for talk given at MIT. 2019. URL: `https://calendar.mit.edu/event/yoshua_bengio_learning_high-level_representations_for_agents#.XYozli2ZNhF`.

[Ber+00] Marcelo Bertalmio et al. "Image inpainting". In: *Computer graphics and interactive techniques*. 2000.

[Bha+17] Arjun Nitin Bhagoji et al. "Exploring the Space of Black-box Attacks on Deep Neural Networks". In: *ArXiv preprint arXiv:1712.09491*. 2017.

[Big+13] Battista Biggio et al. "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases (ECML-KDD)*. 2013.

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning attacks against support vector machines". In: *ICML*. 2012.

[BPR18] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. "Adversarial examples from computational constraints". In: *arXiv preprint arXiv:1805.10204*. 2018.

[BR18] Battista Biggio and Fabio Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In: 2018.

[BSH12] Harold C Burger, Christian J Schuler, and Stefan Harmeling. "Image denoising: Can plain neural networks compete with BM3D?" In: *computer vision and pattern recognition (CVPR)*. 2012.

[Car+19] Nicholas Carlini et al. "On Evaluating Adversarial Robustness". In: *ArXiv preprint arXiv:1902.06705*. 2019.

[CH18] Wengling Chen and James Hays. "Sketchygan: Towards diverse and realistic sketch to image synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[Cha+19] Chun-Hao Chang et al. "Explaining Image Classifiers by Counterfactual Generation". In: *International Conference on Learning Representations (ICLR)*. 2019.

[Che+17a] Pin-Yu Chen et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models". In: *Workshop on Artificial Intelligence and Security*. 2017.

[Che+17b] Xinyun Chen et al. "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning". In: *arXiv preprint arXiv:1712.05526* (2017).

[CRP19] Zachary Charles, Harrison Rosenberg, and Dimitris Papailiopoulos. "A Geometric Perspective on the Transferability of Adversarial Directions". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[CSV17] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. "Learning from untrusted data". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2017, pp. 47–60.

[CW08] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[CW17a] Nicholas Carlini and David Wagner. "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods". In: *Workshop on Artificial Intelligence and Security (AISec)*. 2017.

[CW17b] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *Symposium on Security and Privacy (SP)*. 2017.

[Dan67] John M. Danskin. *The Theory of Max-Min and its Application to Weapons Allocation Problems*. 1967.

[Das+18] Constantinos Daskalakis et al. "Training GANs with Optimism". In: *International Conference on Learning Representations (ICLR)*. 2018.

[Das+19] Constantinos Daskalakis et al. "Efficient Statistics, in High Dimensions, from Truncated Samples". In: *Foundations of Computer Science (FOCS)*. 2019.

[DB16a] Alexey Dosovitskiy and Thomas Brox. "Generating images with perceptual similarity metrics based on deep networks". In: *neural information processing systems (NeurIPS)*. 2016.

[DB16b] Alexey Dosovitskiy and Thomas Brox. "Inverting visual representations with convolutional networks". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[DFE07] Kostadin Dabov, Alessandro Foi, and Karen Egiazarian. "Video denoising by sparse 3D transform-domain collaborative filtering". In: *European Signal Processing Conference*. 2007.

[DG17] Piotr Dabkowski and Yarin Gal. "Real time image saliency for black box classifiers". In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[Dia+16] Ilias Diakonikolas et al. "Robust estimators in high dimensions without the computational intractability". In: *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*. IEEE. 2016, pp. 655–664.

[Dia+17] Ilias Diakonikolas et al. "Being Robust (in High Dimensions) Can Be Practical". In: *International Conference on Machine Learning*. 2017, pp. 999–1008.

[Dia+18] Ilias Diakonikolas et al. "Sever: A Robust Meta-Algorithm for Stochastic Optimization". In: *arXiv preprint arXiv:1803.02815* (2018).

[Din+19] Gavin Weiguang Ding et al. "On the Sensitivity of Adversarial Robustness to Input Data Distributions". In: *International Conference on Learning Representations*. 2019.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation". In: *arXiv preprint arXiv:1410.8516*. 2014.

[Don+14] Jeff Donahue et al. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *International conference on machine learning (ICML)*. 2014.

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real NVP". In: *International Conference on Learning Representations (ICLR)*. 2017.

[EL99] Alexei A Efros and Thomas K Leung. "Texture synthesis by non-parametric sampling". In: *conference on computer vision (CVPR)*. 1999.

[Eng+19a] Logan Engstrom et al. "Exploring the Landscape of Spatial Robustness". In: *International Conference on Machine Learning (ICML)*. 2019.

[Eng+19b] Logan Engstrom et al. "Learning Perceptually-Aligned Representations via Adversarial Robustness". In: *ArXiv preprint arXiv:1906.00945*. 2019.

[FF15] Alhussein Fawzi and Pascal Frossard. "Manitest: Are classifiers really invariant?" In: *British Machine Vision Conference (BMVC)*. 2015.

[FFF18] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. "Adversarial vulnerability for any classifier". In: *Advances in Neural Information Processing Systems (NeuRIPS)*. 2018.

[For+19] Nic Ford et al. "Adversarial Examples Are a Natural Consequence of Test Error in Noise". In: *arXiv preprint arXiv:1901.10513*. 2019.

[Fuk80] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* (1980).

[Fur+18] Tommaso Furlanello et al. "Born-Again Neural Networks". In: *International Conference on Machine Learning (ICML)*. 2018.

[FV17] Ruth C Fong and Andrea Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation". In: *International Conference on Computer Vision (ICCV)*. 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. "Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". In: *arXiv preprint arXiv:1708.06733* (2017).

[GEB16] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks". In: *computer vision and pattern recognition (CVPR)*. 2016.

[Gei+19] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." In: *International Conference on Learning Representations*. 2019.

[Gil+18] Justin Gilmer et al. "Adversarial spheres". In: *Workshop of International Conference on Learning Representations (ICLR)*. 2018.

[Gir+14] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *computer vision and pattern recognition (CVPR)*. 2014, pp. 580–587.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech Recognition with Deep Recurrent Neural Networks". In: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2013.

[Goh19a] Gabriel Goh. "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Robust Feature Leakage". In: *Distill* (2019). DOI: 10.23915/distill.00019.2. URL: https://distill.pub/2019/advex-bugs-discussion/response-2.

[Goh19b] Gabriel Goh. "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Two Examples of Useful, Non-Robust Features". In: *Distill* (2019). https://distill.pub/2019/advex-bugs-discussion/response-3. DOI: 10.23915/distill.00019.3.

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets". In: *neural information processing systems (NeurIPS)*. 2014.

[Gra13] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850*. 2013.

[GSS15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations (ICLR)*. 2015.

[Gul+17] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[HD19] Dan Hendrycks and Thomas G. Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations". In: *International Conference on Learning Representations (ICLR)*. 2019.

[He+15] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *international conference on computer vision (ICCV)*. 2015.

[He+16] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[He+17] Warren He et al. “Adversarial example defense: Ensembles of weak defenses are not strong”. In: *USENIX Workshop on Offensive Technologies (WOOT)*. 2017.

[HE07] James Hays and Alexei A Efros. “Scene completion using millions of photographs”. In: *ACM Transactions on Graphics (TOG)*. 2007.

[Her+01] Aaron Hertzmann et al. “Image analogies”. In: *Computer graphics and interactive techniques*. 2001.

[Heu+17] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation*. 1997.

[HVD14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the Knowledge in a Neural Network”. In: *Neural Information Processing Systems (NeurIPS) Deep Learning Workshop*. 2014.

[Ily+18] Andrew Ilyas et al. “Black-box Adversarial Attacks with Limited Queries and Information”. In: *International Conference on Machine Learning (ICML)*. 2018.

[Ily+19] Andrew Ilyas et al. “Adversarial Examples Are Not Bugs, They Are Features”. In: *Neural Information Processing Systems (NeurIPS)*. 2019.

[ISI17] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. “Globally and Locally Consistent Image Completion”. In: *ACM Trans. Graph.* 2017.

[Iso+17] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *conference on computer vision and pattern recognition (CVPR)*. 2017.

[J+15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *neural information processing systems (NeurIPS)*. 2015.

[Jac+19] Jorn-Henrik Jacobsen et al. "Excessive Invariance Causes Adversarial Vulnerability". In: *International Contemporary on Learning Representations*. 2019.

[JAF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision (ECCV)*. 2016.

[JLT18] Saumya Jetley, Nicholas Lord, and Philip Torr. "With friends like these, who needs adversaries?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[JSO18] Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. "i-RevNet: Deep Invertible Networks". In: *International Conference on Learning Representations (ICLR)*. 2018.

[Kar+18] Tero Karras et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*. 2018.

[KD18] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *International Conference on Learning Representations (ICLR)*. 2017.

[KKM18] Adam Klivans, Pravesh K Kothari, and Raghu Meka. "Efficient Algorithms for Outlier-Robust Regression". In: *arXiv preprint arXiv:1803.03241* (2018).

[KL17] Pang Wei Koh and Percy Liang. "Understanding Black-box Predictions via Influence Functions". In: *ICML*. 2017.

[KMF18] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Geometric robustness of deep networks: analysis and improvement". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Kri09] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *Technical report*. 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[KW15] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR)*. 2015.

[Lar+16] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *International Conference on Machine Learning (ICML)*. 2016.

[LDG18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. "Fine-pruning: Defending against backdooring attacks on deep neural networks". In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. 2018.

[LeC98] Yann LeCun. "The MNIST database of handwritten digits". In: *Technical report*. 1998.

[Liu+17] Yanpei Liu et al. "Delving into Transferable Adversarial Examples and Black-box Attacks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[LM00] Beatrice Laurent and Pascal Massart. "Adaptive estimation of a quadratic functional by model selection". In: *Annals of Statistics*. 2000.

[LRV16] Kevin A Lai, Anup B Rao, and Santosh Vempala. "Agnostic estimation of mean and covariance". In: *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*. IEEE. 2016, pp. 665–674.

[Mad+18] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *International Conference on Learning Representations (ICLR)*. 2018.

[MDM18] 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 (AAAI)*. 2018.

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: a simple and accurate method to fool deep neural networks". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Mik+13] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[Miy+18] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *International Conference on Learning Representations (ICLR)*. 2018.

[Moo+17] Seyed-Mohsen Moosavi-Dezfooli et al. "Universal adversarial perturbations". In: *conference on computer vision and pattern recognition (CVPR)*. 2017.

[Mor+18] Alexander Mordvintsev et al. "Differentiable Image Parameterizations". In: *Distill*. 2018.

[MOT15] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going deeper into neural networks*. 2015. URL: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

[MV15] Aravindh Mahendran and Andrea Vedaldi. "Understanding deep image representations by inverting them". In: *computer vision and pattern recognition (CVPR)*. 2015.

[MZ15] Shike Mei and Xiaojin Zhu. "The security of latent dirichlet allocation". In: *Artificial Intelligence and Statistics*. 2015.

[Nak19] Preetum Nakkiran. "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Adversarial Examples are Just Bugs, Too". In: *Distill* (2019). https://distill.pub/2019/advex-bugs-discussion/response-5. DOI: 10.23915/distill.00019.5.

[Ngu+16] Anh Nguyen et al. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *Neural Information Processing Systems (NeurIPS)*. 2016.

[Ngu+17] Anh Nguyen et al. "Plug & play generative networks: Conditional iterative generation of images in latent space". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Conference on computer vision and pattern recognition (CVPR)*. 2015.

[NZP18] Weili Nie, Yang Zhang, and Ankit Patel. "A Theoretical Explanation for Perplexing Behaviors of Backpropagation-based Visualizations". In: *International Conference on Machine Learning (ICML)*. 2018.

[Ola+18] Chris Olah et al. "The Building Blocks of Interpretability". In: *Distill*. 2018.

[OMS17] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill*. 2017.

[Oyg15] Audun Oygard. *Visualizing GoogLeNet Classes*. 2015. URL: https://www.auduno.com/2015/07/29/visualizing-googlenet-classes/.

[Pap+17] Nicolas Papernot et al. "Practical black-box attacks against machine learning". In: *Asia Conference on Computer and Communications Security*. 2017.

[Par+19] Taesung Park et al. "Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *Computer Vision and Pattern Recognition (CVPR)*. 2019.

[PMG16] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in Machine Learning: from Phenomena to Black-box Attacks using Adversarial Samples". In: *ArXiv preprint arXiv:1605.07277*. 2016.

[Pra+18] Adarsh Prasad et al. "Robust Estimation via Robust Gradient Estimation". In: *arXiv preprint arXiv:1802.06485* (2018).

[Rec+19] Benjamin Recht et al. "Do CIFAR-10 Classifiers Generalize to CIFAR-10?" In: *International Conference on Machine Learning (ICML)*. 2019.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International Conference on Machine Learning (ICML)*. 2015.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *International Conference on Learning Representations (ICLR)*. 2016.

[Rus+15] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)*. 2015.

[Sal+16] Tim Salimans et al. "Improved techniques for training gans". In: *neural information processing systems (NeurIPS)*. 2016.

[Sch+18] Ludwig Schmidt et al. "Adversarially Robust Generalization Requires More Data". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[Sha+16] Mahmood Sharif et al. "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. 2016, pp. 1528–1540.

[Sha+18] Ali Shafahi et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[Sha+19a] Ali Shafahi et al. "Are adversarial examples inevitable?" In: *International Conference on Learning Representations (ICLR)*. 2019.

[Sha+19b] Adi Shamir et al. "A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance". In: *arXiv preprint arXiv:1901.10861*. 2019.

[SHS19] David Stutz, Matthias Hein, and Bernt Schiele. "Disentangling Adversarial Robustness and Generalization". In: *Computer Vision and Pattern Recognition (CVPR)*. 2019.

[SKL17] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. "Certified Defenses for Data Poisoning Attacks". In: *NIPS*. 2017.

[Smi+17] D. Smilkov et al. "SmoothGrad: removing noise by adding noise". In: *ICML workshop on visualization for deep learning*. 2017.

[SSA18] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. "How good is my GAN?" In: *European Conference on Computer Vision (ECCV)*. 2018.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International Conference on Machine Learning (ICML)*. 2017.

[Su+18] Dong Su et al. "Is Robustness the Cost of Accuracy? A Comprehensive Study on the Robustness of 18 Deep Image Classification Models". In: *European Conference on Computer Vision (ECCV)*. 2018.

[Sug+19] Arun Sai Suggala et al. "Revisiting Adversarial Risk". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)*. 2015.

[Sze+14] Christian Szegedy et al. "Intriguing properties of neural networks". In: *International Conference on Learning Representations (ICLR)*. 2014.

[Tai+14] Yaniv Taigman et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *Computer Vision and Pattern Recognition (CVPR)*. 2014.

[TB17] Florian Tramèr and Dan Boneh. *Personal Communication*. 2017.

[TG16] Thomas Tanay and Lewis Griffin. "A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples". In: *ArXiv preprint arXiv:1608.07690*. 2016.

[Tra+17] Florian Tramer et al. "The Space of Transferable Adversarial Examples". In: *ArXiv preprint arXiv:1704.03453*. 2017.

[Tsi+19] Dimitris Tsipras et al. "Robustness May Be at Odds with Accuracy". In: *International Conference on Learning Representations (ICLR)*. 2019.

[Tyk16] Mike Tyka. *Class visualization with bilateral filters*. 2016. URL: `https://mtyka.github.io/deepdream/2016/02/05/bilateral-class-vis.html`.

[Ues+18] Jonathan Uesato et al. "Adversarial Risk and the Dangers of Evaluating Against Weak Attacks". In: *International Conference on Machine Learning (ICML)*. 2018.

[Upc+17] Paul Upchurch et al. "Deep feature interpolation for image content changes". In: *conference on computer vision and pattern recognition (CVPR)*. 2017.

[UVL17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep Image Prior". In: *ArXiv preprint arXiv:1711.10925*. 2017.

[VCL19] Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. "Deep learning generalizes because the parameter-function map is biased towards simple functions". In: *International Conference on Learning Representations*. 2019.

[Ver18] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. 2018.

[Vin+10] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research (JMLR)*. 2010.

[VKK16] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: *International Conference on Machine Learning (ICML)*. 2016.

[Wal45] Abraham Wald. "Statistical Decision Functions Which Minimize the Maximum Risk". In: *Annals of Mathematics*. 1945.

[Wan+18] Tongzhou Wang et al. "Dataset Distillation". In: *ArXiv preprint arXiv:1811.10959*. 2018.

[Wu+16] Yuxin Wu et al. *Tensorpack*. `https://github.com/tensorpack/`. 2016.

[Xia+15] Huang Xiao et al. "Support vector machines under adversarial label contamination". In: *Neurocomputing*. Vol. 160. 2015, pp. 53–62.

[Xia+18] Chaowei Xiao et al. "Spatially Transformed Adversarial Examples". In: *International Conference on Learning Representations (ICLR)*. 2018.

[Yeh+17] Raymond A Yeh et al. "Semantic image inpainting with deep generative models". In: *Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Yos+15] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579*. 2015.

[Yu+18] Jiahui Yu et al. "Generative image inpainting with contextual attention". In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Zha+18a] Han Zhang et al. "Self-attention generative adversarial networks". In: *arXiv preprint arXiv:1805.08318*. 2018.

[Zha+18b] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Zhu+17] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *international conference on computer vision(ICCV)*. 2017.

[Zou+18] Haosheng Zou et al. "Geometric Universality of Adversarial Examples in Deep Learning". In: *Geometry in Machine Learning ICML Workshop (GIML)*. 2018.

[ZZ19] Tianyuan Zhang and Zhanxing Zhu. "Interpreting Adversarially Trained Convolutional Neural Networks". In: *International Conference on Machine Learning (ICML)*. 2019.

Appendix A

Appendix

A.1 Gaussian MLE under Adversarial Perturbation

In this section, we develop a framework for studying non-robust features by studying the problem of *maximum likelihood classification* between two Gaussian distributions. We first recall the setup of the problem, then present the main theorems from Section 2.6. First we build the techniques necessary for their proofs.

A.1.1 Setup

We consider the setup where a learner receives labeled samples from two distributions, $\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$, and $\mathcal{N}(-\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$. The learner's goal is to be able to classify new samples as being drawn from $\mathcal{D}_1$ or $\mathcal{D}_2$ according to a maximum likelihood (MLE) rule.

A simple coupling argument demonstrates that this problem can actually be reduced to learning the parameters $\widehat{\boldsymbol{\mu}}$, $\widehat{\boldsymbol{\Sigma}}$ of a single Gaussian $\mathcal{N}(-\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$, and then employing a linear classifier with weight $\widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}$. In the standard setting, maximum likelihoods estimation learns the true parameters, $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$, and thus the learned classification rule is $C(x) = \mathbb{1}\{x^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} > 0\}$.

Here, we consider the problem of *adversarially robust* maximum likelihood estimation. In particular, rather than simply being asked to classify samples, the learner

will be asked to classify *adversarially perturbed* samples $x + \delta$, where $\delta \in \Delta$ is chosen to maximize the loss of the learner. Our goal is to derive the parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ corresponding to an adversarially robust maximum likelihood estimate of the parameters of $\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$. Note that since we have access to $\boldsymbol{\Sigma}_*$ (indeed, the learner can just run non-robust MLE to get access), we work in the space where $\boldsymbol{\Sigma}^*$ is a diagonal matrix, and we restrict the learned covariance $\boldsymbol{\Sigma}$ to the set of diagonal matrices.

Notation. We denote the parameters of the sampled Gaussian by $\boldsymbol{\mu}_* \in \mathbb{R}^d$, and $\boldsymbol{\Sigma}_* \in \{\text{diag}(\boldsymbol{u}) | \boldsymbol{u} \in \mathbb{R}^d\}$. We use $\sigma_{min}(X)$ to represent the smallest eigenvalue of a square matrix X, and $\ell(\cdot; x)$ to represent the Gaussian negative log-likelihood for a single sample x. For convenience, we often use $\boldsymbol{v} = x - \boldsymbol{\mu}$, and $R = \|\boldsymbol{\mu}_*\|$. We also define the $\oslash$ operator to represent the vectorization of the diagonal of a matrix. In particular, for a matrix $X \in \mathbb{R}^{d \times d}$, we have that $X_{\oslash} = v \in \mathbb{R}^d$ if $v_i = X_{ii}$.

A.1.2 Outline and Key Results

We focus on the case where $\Delta = \mathcal{B}_2(\varepsilon)$ for some $\varepsilon > 0$, i.e. the ℓ_2 ball, corresponding to the following minimax problem:

$$\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\max_{\delta : \|\delta\| = \varepsilon} \ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x + \delta) \right] \tag{A.1}$$

We first derive the optimal adversarial perturbation for this setting (Section A.1.3), and prove Theorem 1 (Section A.1.3). We then propose an alternate problem, in which the adversary picks a linear operator to be applied to a fixed vector, rather than picking a specific perturbation vector (Section A.1.3). We argue via Gaussian concentration that the alternate problem is indeed reflective of the original model (and in particular, the two become equivalent as $d \to \infty$). In particular, we propose studying the following in place of (A.1):

$$\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} \max_{M \in \mathcal{M}} \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x + M(x - \boldsymbol{\mu})) \right] \tag{A.2}$$

$$\text{where } \mathcal{M} = \left\{ M \in \mathbb{R}^{d \times d} : \ M_{ij} = 0 \ \forall \ i \neq j, \ \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\|M \boldsymbol{v}\|_2^2 \right] = \varepsilon^2 \right\}.$$

Our goal is to characterize the behavior of the robustly learned covariance $\mathbf{\Sigma}$ in terms of the true covariance matrix $\mathbf{\Sigma}_*$ and the perturbation budget ε. The proof is through Danskin's Theorem, which allows us to use any maximizer of the inner problem M^* in computing the subgradient of the inner minimization. After showing the applicability of Danskin's Theorem (Section A.1.3) and then applying it (Section A.1.3) to prove our main results (Section A.1.3). Our three main results, which we prove in the following section, are presented below.

First, we consider a simplified version of (A.1), in which the adversary solves a maximization with a fixed Lagrangian penalty, rather than a hard ℓ_2 constraint. In this setting, we show that the loss contributed by the adversary corresponds to a misalignment between the data metric (the Mahalanobis distance, induced by $\mathbf{\Sigma}^{-1}$), and the ℓ_2 metric:

Theorem 1 (Adversarial vulnerability from misalignment)**.** *Consider an adversary whose perturbation is determined by the "Lagrangian penalty" form of* (2.14)*, i.e.*

$$\max_{\delta} \ell(x+\delta; y \cdot \boldsymbol{\mu}, \mathbf{\Sigma}) - C \cdot \|\delta\|_2,$$

where $C \geq \frac{1}{\sigma_{min}(\mathbf{\Sigma}_*)}$ *is a constant trading off NLL minimization and the adversarial constraint*[1]*. Then, the adversarial loss* $\mathcal{L}_{adv}$ *incurred by the non-robustly learned* $(\boldsymbol{\mu}, \mathbf{\Sigma})$ *is given by:*

$$\mathcal{L}_{adv}(\Theta) - \mathcal{L}(\Theta) = tr\left[\left(I + (C \cdot \mathbf{\Sigma}_* - I)^{-1}\right)^2\right] - d,$$

and, for a fixed $tr(\mathbf{\Sigma}_*) = k$ *the above is minimized by* $\mathbf{\Sigma}_* = \frac{k}{d}\boldsymbol{I}$*.*

We then return to studying (A.2), where we provide upper and lower bounds on the learned robust covariance matrix $\mathbf{\Sigma}$:

Theorem 2 (Robustly Learned Parameters)**.** *Just as in the non-robust case,* $\boldsymbol{\mu}_r = \boldsymbol{\mu}^*$*, i.e. the true mean is learned. For the robust covariance* $\mathbf{\Sigma}_r$*, there exists an* $\varepsilon_0 > 0$*,*

[1]The constraint on C is to ensure the problem is concave.

such that for any $\varepsilon \in [0, \varepsilon_0)$,

$$\boldsymbol{\Sigma}_r = \frac{1}{2}\boldsymbol{\Sigma}_* + \frac{1}{\lambda} \cdot \boldsymbol{I} + \sqrt{\frac{1}{\lambda} \cdot \boldsymbol{\Sigma}_* + \frac{1}{4}\boldsymbol{\Sigma}_*^2}, \quad \textit{where} \quad \Omega\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}+\varepsilon^{3/2}}\right) \leq \lambda \leq O\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}}\right).$$

Finally, we show that in the worst case over mean vectors $\boldsymbol{\mu}_*$, the gradient of the adversarial robust classifier aligns more with the inter-class vector:

Theorem 3 (Gradient alignment)**.** *Let $f(x)$ and $f_r(x)$ be monotonic classifiers based on the linear separator induced by standard and ℓ_2-robust maximum likelihood classification, respectively. The maximum angle formed between the gradient of the classifier (wrt input) and the vector connecting the classes can be smaller for the robust model:*

$$\min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f_r(x) \rangle}{\|\boldsymbol{\mu}\| \cdot \|\nabla_x f_r(x)\|} > \min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f(x) \rangle}{\|\boldsymbol{\mu}\| \cdot \|\nabla_x f(x)\|}.$$

A.1.3 Proofs

In the first section, we have shown that the classification between two Gaussian distributions with identical covariance matrices centered at $\boldsymbol{\mu}^*$ and $-\boldsymbol{\mu}^*$ can in fact be reduced to learning the parameters of a single one of these distributions.

Thus, in the standard setting, our goal is to solve the following problem:

$$\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)} \left[\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x)\right] := \min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)} \left[-\log\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x)\right)\right].$$

Note that in this setting, one can simply find differentiate ℓ with respect to both $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and obtain closed forms for both (indeed, these closed forms are, unsurprisingly, $\boldsymbol{\mu}^*$ and $\boldsymbol{\Sigma}^*$). Here, we consider the existence of a *malicious adversary* who is allowed to perturb each sample point x by some δ. The goal of the adversary is to *maximize* the same loss that the learner is minimizing.

Motivating example: ℓ_2-constrained adversary

We first consider, as a motivating example, an ℓ_2-constrained adversary. That is, the adversary is allowed to perturb each sampled point by $\delta : \|\delta\|_2 = \varepsilon$. In this case, the

minimax problem being solved is the following:

$$\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)} \left[\max_{\|\delta\|=\varepsilon} \ell(\boldsymbol{\mu},\boldsymbol{\Sigma}; x+\delta) \right]. \tag{A.3}$$

The following Lemma captures the optimal behaviour of the adversary:

Lemma 4. *In the minimax problem captured in* (A.3) *(and earlier in* (A.1)*), the optimal adversarial perturbation δ^* is given by*

$$\delta^* = \left(\lambda \boldsymbol{I} - \boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1}\boldsymbol{v} = \left(\lambda\boldsymbol{\Sigma} - \boldsymbol{I}\right)^{-1} \boldsymbol{v}, \tag{A.4}$$

where $\boldsymbol{v} = x - \boldsymbol{\mu}$, and λ is set such that $\|\delta^*\|_2 = \varepsilon$.

Proof. In this context, we can solve the inner maximization problem with Lagrange multipliers. In the following we write $\Delta = \mathcal{B}_2(\varepsilon)$ for brevity, and discard terms not containing δ as well as constant factors freely:

$$\begin{aligned}
\arg\max_{\delta\in\Delta} \ \ell(\boldsymbol{\mu},\boldsymbol{\Sigma}; x+\delta) &= \arg\max_{\delta\in\Delta} \left(x+\delta-\boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1} \left(x+\delta-\boldsymbol{\mu}\right) \\
&= \arg\max_{\delta\in\Delta} \ (x-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) + 2\delta^\top\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) + \delta^\top\boldsymbol{\Sigma}^{-1}\delta \\
&= \arg\max_{\delta\in\Delta} \ \delta^\top\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) + \frac{1}{2}\delta^\top\boldsymbol{\Sigma}^{-1}\delta.
\end{aligned} \tag{A.5}$$

Now we can solve (A.5) using the aforementioned Lagrange multipliers. In particular, note that the maximum of (A.5) is attained at the boundary of the ℓ_2 ball Δ. Thus, we can solve the following system of two equations to find δ, rewriting the norm constraint as $\frac{1}{2}\|\delta\|_2^2 = \frac{1}{2}\varepsilon^2$:

$$\begin{aligned}
&\nabla_\delta \left(\delta^\top\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) + \frac{1}{2}\delta^\top\boldsymbol{\Sigma}^{-1}\delta\right) = \lambda\nabla_\delta\left(\|\delta\|_2^2 - \varepsilon^2\right) \implies \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}) + \boldsymbol{\Sigma}^{-1}\delta = \lambda\delta \\
&\|\delta\|_2^2 = \varepsilon^2.
\end{aligned} \tag{A.6}$$

For clarity, we write $\boldsymbol{v} = x - \boldsymbol{\mu}$: then, combining the above, we have that

$$\delta^* = \left(\lambda \boldsymbol{I} - \boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1} \boldsymbol{v} = \left(\lambda \boldsymbol{\Sigma} - \boldsymbol{I}\right)^{-1} \boldsymbol{v}, \tag{A.7}$$

our final result for the maximizer of the inner problem, where λ is set according to the norm constraint. □

Variant with Fixed Lagrangian (Theorem 1)

To simplify the analysis of Theorem 1, we consider a version of (A.3) with a fixed Lagrangian penalty, rather than a norm constraint:

$$\max \ell(x + \delta; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma}) - C \cdot \|\delta\|_2.$$

Note then, that by Lemma 4, the optimal perturbation δ^* is given by

$$\delta^* = \left(C\boldsymbol{\Sigma} - \boldsymbol{I}\right)^{-1}.$$

We now proceed to the proof of Theorem 1.

Theorem 1 (Adversarial vulnerability from misalignment)**.** *Consider an adversary whose perturbation is determined by the "Lagrangian penalty" form of* (2.14)*, i.e.*

$$\max_{\delta} \ell(x + \delta; y \cdot \boldsymbol{\mu}, \boldsymbol{\Sigma}) - C \cdot \|\delta\|_2,$$

where $C \geq \frac{1}{\sigma_{min}(\boldsymbol{\Sigma}_*)}$ *is a constant trading off NLL minimization and the adversarial constraint*[2]*. Then, the adversarial loss* $\mathcal{L}_{adv}$ *incurred by the non-robustly learned* $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *is given by:*

$$\mathcal{L}_{adv}(\Theta) - \mathcal{L}(\Theta) = tr\left[\left(I + (C \cdot \boldsymbol{\Sigma}_* - I)^{-1}\right)^2\right] - d,$$

and, for a fixed $tr(\boldsymbol{\Sigma}_*) = k$ *the above is minimized by* $\boldsymbol{\Sigma}_* = \frac{k}{d}\boldsymbol{I}$*.*

[2]The constraint on C is to ensure the problem is concave.

Proof. We begin by expanding the Gaussian negative log-likelihood for the relaxed problem:

$$\begin{aligned}
\mathcal{L}_{adv}(\Theta) - \mathcal{L}(\Theta) &= \mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)}\Big[2\cdot\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}-\boldsymbol{I}\right)^{-\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{v} \\
&\qquad +\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}-\boldsymbol{I}\right)^{-\top}\boldsymbol{\Sigma}^{-1}\left(C\cdot\boldsymbol{\Sigma}-\boldsymbol{I}\right)^{-1}\boldsymbol{v}\Big] \\
&= \mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)}\big[2\cdot\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}\boldsymbol{\Sigma}-\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{v} \\
&\qquad +\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}-\boldsymbol{I}\right)^{-\top}\boldsymbol{\Sigma}^{-1}\left(C\cdot\boldsymbol{\Sigma}-\boldsymbol{I}\right)^{-1}\boldsymbol{v}\Big]
\end{aligned}$$

Recall that we are considering the vulnerability at the MLE parameters $\boldsymbol{\mu}^*$ and $\boldsymbol{\Sigma}^*$:

$$\begin{aligned}
\mathcal{L}_{adv}(\Theta) - \mathcal{L}(\Theta) &= \mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,I)}\Big[2\cdot\boldsymbol{v}^\top\boldsymbol{\Sigma}_*^{1/2}\left(C\cdot\boldsymbol{\Sigma}_*^2-\boldsymbol{\Sigma}_*\right)^{-1}\boldsymbol{\Sigma}_*^{1/2}\boldsymbol{v} \\
&\qquad +\boldsymbol{v}^\top\boldsymbol{\Sigma}_*^{1/2}\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-\top}\boldsymbol{\Sigma}_*^{-1}\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\boldsymbol{\Sigma}_*^{1/2}\boldsymbol{v}\Big] \\
&= \mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,I)}\big[2\cdot\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\boldsymbol{v} \\
&\qquad +\boldsymbol{v}^\top\boldsymbol{\Sigma}_*^{1/2}\left(C^2\boldsymbol{\Sigma}_*^3-2C\cdot\boldsymbol{\Sigma}_*^2+\boldsymbol{\Sigma}_*\right)^{-1}\boldsymbol{\Sigma}_*^{1/2}\boldsymbol{v}\Big] \\
&= \mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,I)}\big[2\cdot\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\boldsymbol{v} \\
&\qquad +\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-2}\boldsymbol{v}\big] \\
&= \mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,I)}\big[-\|v\|_2^2+\boldsymbol{v}^\top\boldsymbol{I}\boldsymbol{v} \\
&\qquad +2\cdot\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\boldsymbol{v}+\boldsymbol{v}^\top\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-2}\boldsymbol{v}\big] \\
&= \mathbb{E}_{\boldsymbol{v}\sim\mathcal{N}(0,I)}\left[-\|v\|_2^2+\boldsymbol{v}^\top\left(\boldsymbol{I}+\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\right)^2\boldsymbol{v}\right] \\
&= \operatorname{tr}\left[\left(\boldsymbol{I}+\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\right)^2\right]-d
\end{aligned}$$

This shows the first part of the theorem. It remains to show that for a fixed $k=\operatorname{tr}(\boldsymbol{\Sigma}_*)$, the adversarial risk is minimized by $\boldsymbol{\Sigma}_*=\frac{k}{d}\boldsymbol{I}$:

$$\begin{aligned}
\min_{\boldsymbol{\Sigma}_*}\ \mathcal{L}_{adv}(\Theta)-\mathcal{L}(\Theta) &= \min_{\boldsymbol{\Sigma}_*}\ \operatorname{tr}\left[\left(\boldsymbol{I}+\left(C\cdot\boldsymbol{\Sigma}_*-\boldsymbol{I}\right)^{-1}\right)^2\right] \\
&= \min_{\{\sigma_i\}}\ \sum_{i=1}^{d}\left(1+\frac{1}{C\cdot\sigma_i-1}\right)^2,
\end{aligned}$$

where $\{\sigma_i\}$ are the eigenvalues of $\boldsymbol{\Sigma}_*$. Now, we have that $\sum\sigma_i=k$ by assumption, so

by optimality conditions, we have that $\boldsymbol{\Sigma}_*$ minimizes the above if $\nabla_{\{\sigma_i\}} \propto \vec{1}$, i.e. if $\nabla_{\sigma_i} = \nabla_{\sigma_j}$ for all i, j. Now,

$$\begin{aligned}\nabla_{\sigma_i} &= -2 \cdot \left(1 + \frac{1}{C \cdot \sigma_i - 1}\right) \cdot \frac{C}{(C \cdot \sigma_i - 1)^2} \\ &= -2 \cdot \frac{C^2 \cdot \sigma_i}{(C \cdot \sigma_i - 1)^3}.\end{aligned}$$

Then, by solving analytically, we find that

$$-2 \cdot \frac{C^2 \cdot \sigma_i}{(C \cdot \sigma_i - 1)^3} = -2 \cdot \frac{C^2 \cdot \sigma_j}{(C \cdot \sigma_j - 1)^3}$$

admits only one real solution, $\sigma_i = \sigma_j$. Thus, $\boldsymbol{\Sigma}_* \propto \boldsymbol{I}$. Scaling to satisfy the trace constraint yields $\boldsymbol{\Sigma}_* = \frac{k}{d}\boldsymbol{I}$, which concludes the proof. □

Real objective

Our motivating example (Section A.1.3) demonstrates that the optimal perturbation for the adversary in the ℓ_2-constrained case is actually a linear function of $\boldsymbol{v}$, and in particular, that the optimal perturbation can be expressed as $D\boldsymbol{v}$ for a diagonal matrix D. Note, however, that the problem posed in (A.3) is not actually a minimax problem, due to the presence of the expectation between the outer minimization and the inner maximization. Motivated by this and (A.7), we define the following robust problem:

$$\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \max_{M \in \mathcal{M}} \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x + M\boldsymbol{v})\right], \tag{A.8}$$

$$\text{where } \mathcal{M} = \left\{M \in \mathbb{R}^{d \times d} : \; M_{ij} = 0 \; \forall \; i \neq j, \; \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\|M\boldsymbol{v}\|_2^2\right] = \varepsilon^2\right\}.$$

First, note that this objective is slightly different from that of (A.3). In the motivating example, δ is constrained to *always* have ε-norm, and thus is normalizer on a per-sample basis inside of the expectation. In contrast, here the classifier is concerned with being robust to perturbations that are linear in $\boldsymbol{v}$, and of ε^2 squared norm *in expectation*.

Note, however, that via the result of Laurent and Massart [LM00] showing strong concentration for the norms of Gaussian random variables, in high dimensions this bound on expectation has a corresponding high-probability bound on the norm. In particular, this implies that as $d \to \infty$, $\|M\boldsymbol{v}\|_2 = \varepsilon$ almost surely, and thus the problem becomes identical to that of (A.3). We now derive the optimal M for a given $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$:

Lemma 5. *Consider the minimax problem described by* (A.8)*, i.e.*

$$\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \max_{M \in \mathcal{M}} \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}; x + M\boldsymbol{v})\right].$$

Then, the optimal action M^* of the inner maximization problem is given by

$$M = (\lambda \boldsymbol{\Sigma} - \boldsymbol{I})^{-1}, \tag{A.9}$$

where again λ is set so that $M \in \mathcal{M}$.

Proof. We accomplish this in a similar fashion to what was done for δ^*, using Lagrange multipliers:

$$\begin{aligned}
\nabla_M \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\boldsymbol{v}^\top M \boldsymbol{\Sigma}^{-1} \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^\top M \boldsymbol{\Sigma}^{-1} M \boldsymbol{v}\right] &= \lambda \nabla_M \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\|M\boldsymbol{v}\|_2^2 - \varepsilon^2\right] \\
\mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\boldsymbol{\Sigma}^{-1} \boldsymbol{v}\boldsymbol{v}^\top + \boldsymbol{\Sigma}^{-1} M \boldsymbol{v}\boldsymbol{v}^\top\right] &= \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\lambda M \boldsymbol{v}\boldsymbol{v}^\top\right] \\
\boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}^* + \boldsymbol{\Sigma}^{-1} M \boldsymbol{\Sigma}^* &= \lambda M \boldsymbol{\Sigma}^* \\
M &= (\lambda \boldsymbol{\Sigma} - \boldsymbol{I})^{-1},
\end{aligned}$$

where λ is a constant depending on $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ enforcing the expected squared-norm constraint. □

Indeed, note that the optimal M for the adversary takes a near-identical form to the optimal δ (A.7), with the exception that λ is not sample-dependent but rather varies only with the parameters.

Danskin's Theorem

The main tool in proving our key results is Danskin's Theorem [Dan67], a powerful theorem from minimax optimization which contains the following key result:

Theorem 4 (Danskin's Theorem)**.** *Suppose $\phi(x, z) : \mathbb{R} \times Z \to \mathbb{R}$ is a continuous function of two arguments, where $Z \subset \mathbb{R}^m$ is compact. Define $f(x) = \max_{z \in Z} \phi(x, z)$. Then, if for every $z \in Z$, $\phi(x, z)$ is convex and differentiable in x, and $\frac{\partial \phi}{\partial x}$ is continuous:*

The subdifferential of $f(x)$ is given by

$$\partial f(x) = \operatorname{conv}\left\{\frac{\partial \phi(x, z)}{\partial x} : z \in Z_0(x)\right\},$$

where $\operatorname{conv}(\cdot)$ *represents the convex hull operation, and Z_0 is the set of maximizers defined as*

$$Z_0(x) = \left\{\overline{z} : \phi(x, \overline{z}) = \max_{z \in Z} \phi(x, z)\right\}.$$

In short, given a minimax problem of the form $\min_x \max_{y \in C} f(x, y)$ where C is a compact set, if $f(\cdot, y)$ is convex for all values of y, then rather than compute the gradient of $g(x) := \max_{y \in C} f(x, y)$, we can simply find a maximizer y^* for the current parameter x; Theorem 4 ensures that $\nabla_x f(x, y^*) \in \partial_x g(x)$. Note that $\mathcal{M}$ is trivially compact (by the Heine-Borel theorem), and differentiability/continuity follow rather straightforwardly from our reparameterization (c.f. (A.10)), and so it remains to show that the outer minimization is convex for any fixed M.

Convexity of the outer minimization. Note that even in the standard case (i.e. non-adversarial), the Gaussian negative log-likelihood is not convex with respect to $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Thus, rather than proving convexity of this function directly, we employ the parameterization used by [Das+19]: in particular, we write the problem in terms of $\boldsymbol{T} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{m} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. Under this parameterization, we show that the robust problem is convex for any fixed M.

Lemma 6. *Under the aforementioned parameterization of $\boldsymbol{T} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{m} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$,*

the following "Gaussian robust negative log-likelihood" is convex:

$$\mathbb{E}_{x\sim\mathcal{N}(\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)}\left[\ell(\boldsymbol{m},\boldsymbol{T};x+M\boldsymbol{v})\right].$$

Proof. To prove this, we show that the likelihood is convex even with respect to a single sample x; the result follows, since a convex combination of convex functions remains convex. We begin by looking at the likelihood of a single sample $x\sim\mathcal{N}(\boldsymbol{\mu}_*,\boldsymbol{\Sigma}_*)$:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\mu},\boldsymbol{\Sigma};x+M(x-\boldsymbol{\mu})) &= \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top(I+M)^2\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)\\
&= \frac{\frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top(I+M)^2\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)}{\int\frac{\exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^\top(I+M)^2\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|(\boldsymbol{I}+\boldsymbol{M})^{-2}\boldsymbol{\Sigma}|}}}\\
&= \frac{|I+M|^{-1}\exp\left(-\frac{1}{2}x^\top(I+M)^2\boldsymbol{\Sigma}^{-1}x+\boldsymbol{\mu}^\top(I+M)^2\boldsymbol{\Sigma}^{-1}x\right)}{\int\exp\left(-\frac{1}{2}x^\top(I+M)^2\boldsymbol{\Sigma}^{-1}x+\boldsymbol{\mu}^\top(I+M)^2\boldsymbol{\Sigma}^{-1}x\right)}
\end{aligned}$$

In terms of the aforementioned $\boldsymbol{T}$ and $\boldsymbol{m}$, and for convenience defining $A=(I+M)^2$:

$$\begin{aligned}
\ell(x) &= |A|^{-1/2}+\left(\frac{1}{2}x^\top A\boldsymbol{T}x-\boldsymbol{m}^\top Ax\right)-\log\left(\int\exp\left(\frac{1}{2}x^\top A\boldsymbol{T}x-\boldsymbol{m}^\top Ax\right)\right)\\
\nabla\ell(x) &= \begin{bmatrix}\frac{1}{2}(Axx^\top)_{\circledast}\\ -Ax\end{bmatrix}-\frac{\int\begin{bmatrix}\frac{1}{2}(Axx^\top)_{\circledast}\\ -Ax\end{bmatrix}\exp\left(\frac{1}{2}x^\top A\boldsymbol{T}x-\boldsymbol{m}^\top Ax\right)}{\int\exp\left(\frac{1}{2}x^\top A\boldsymbol{T}x-\boldsymbol{m}^\top Ax\right)}\\
&= \begin{bmatrix}\frac{1}{2}(Axx^\top)_{\circledast}\\ -Ax\end{bmatrix}-\mathbb{E}_{z\sim\mathcal{N}(\boldsymbol{T}^{-1}\boldsymbol{m},(\boldsymbol{AT})^{-1})}\begin{bmatrix}\frac{1}{2}(Azz^\top)_{\circledast}\\ -Az\end{bmatrix}.
\end{aligned}\tag{A.10}$$

From here, following an identical argument to [Das+19] Equation (3.7), we find that

$$\boldsymbol{H}_\ell=\mathrm{Cov}_{z\sim\mathcal{N}(\boldsymbol{T}^{-1}\boldsymbol{m},(A\boldsymbol{T})^{-1})}\left[\begin{pmatrix}(-\frac{1}{2}Azz^T)_{\circledast}\\ z\end{pmatrix},\begin{pmatrix}(-\frac{1}{2}Azz^T)_{\circledast}\\ z\end{pmatrix}\right]\succcurlyeq\mathbf{0},$$

i.e. that the log-likelihood is indeed convex with respect to $\begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{m} \end{bmatrix}$, as desired. $\square$

Applying Danskin's Theorem

The previous two parts show that we can indeed apply Danskin's theorem to the outer minimization, and in particular that the gradient of f at $M = M^*$ is in the subdifferential of the outer minimization problem. We proceed by writing out this gradient explicitly, and then setting it to zero (note that since we have shown f is convex for all choices of perturbation, we can use the fact that a convex function is globally minimized $\leftrightarrow$ its subgradient contains zero). We continue from above, plugging in (A.9) for M and using (A.10) to write the gradients of ℓ with respect to $\boldsymbol{T}$ and $\boldsymbol{m}$.

$$
\begin{aligned}
0 = \nabla_{\begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{m} \end{bmatrix}} \ell &= \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \left[\begin{bmatrix} \frac{1}{2}(Axx^\top)_{\oslash} \\ -Ax \end{bmatrix} - \mathbb{E}_{z \sim \mathcal{N}(\boldsymbol{T}^{-1}\boldsymbol{m}, (\boldsymbol{A}\boldsymbol{T})^{-1})} \begin{bmatrix} \frac{1}{2}(Azz^\top)_{\oslash} \\ -Az \end{bmatrix} \right] \\
&= \mathbb{E}_{x \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)} \begin{bmatrix} \frac{1}{2}(Axx^\top)_{\oslash} \\ -Ax \end{bmatrix} - \mathbb{E}_{z \sim \mathcal{N}(\boldsymbol{T}^{-1}\boldsymbol{m}, (\boldsymbol{A}\boldsymbol{T})^{-1})} \begin{bmatrix} \frac{1}{2}(Azz^\top)_{\oslash} \\ -Az \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2}(A\boldsymbol{\Sigma}_*)_{\oslash} \\ -A\boldsymbol{\mu}_* \end{bmatrix} - \mathbb{E}_{z \sim \mathcal{N}(\boldsymbol{T}^{-1}\boldsymbol{m}, (\boldsymbol{A}\boldsymbol{T})^{-1})} \begin{bmatrix} \frac{1}{2}(A(A\boldsymbol{T})^{-1})_{\oslash} \\ -A\boldsymbol{T}^{-1}\boldsymbol{m} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2}A\boldsymbol{\Sigma}_* \\ -A\boldsymbol{\mu}_* \end{bmatrix} - \begin{bmatrix} \frac{1}{2}A(A\boldsymbol{T})^{-1} \\ -A\boldsymbol{T}^{-1}\boldsymbol{m} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2}A\boldsymbol{\Sigma}_* - \frac{1}{2}\boldsymbol{T}^{-1} \\ A\boldsymbol{T}^{-1}\boldsymbol{m} - A\boldsymbol{\mu}_* \end{bmatrix} \qquad \text{(A.11)}
\end{aligned}
$$

Using this fact, we derive an *implicit* expression for the robust covariance matrix $\boldsymbol{\Sigma}$. Note that for the sake of brevity, we now use M to denote the optimal adversarial perturbation (previously defined as M^* in (A.9)). This implicit formulation forms the foundation of the bounds given by our main results.

Lemma 7. *The minimax problem discussed throughout this work admits the following (implicit) form of solution:*

$$\boldsymbol{\Sigma} = \frac{1}{\lambda} I + \frac{1}{2}\boldsymbol{\Sigma}_* + \sqrt{\frac{1}{\lambda}\boldsymbol{\Sigma}_* + \frac{1}{4}\boldsymbol{\Sigma}_*^2},$$

where λ is such that $M \in \mathcal{M}$, and is thus dependent on $\boldsymbol{\Sigma}$.

Proof. Rewriting (A.11) in the standard parameterization (with respect to $\boldsymbol{\mu}, \boldsymbol{\Sigma}$) and re-expanding $A = (I + M)^2$ yields:

$$0 = \nabla_{\begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{m} \end{bmatrix}} \ell = \begin{bmatrix} \frac{1}{2}(I+M)^2\boldsymbol{\Sigma}_* - \frac{1}{2}\boldsymbol{\Sigma} \\ (I+M)^2\boldsymbol{\mu} - (I+M)^2\boldsymbol{\mu}_* \end{bmatrix}$$

Now, note that the equations involving $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are completely independent, and thus can be solved separately. In terms of $\boldsymbol{\mu}$, the relevant system of equations is $A\boldsymbol{\mu} - A\boldsymbol{\mu}_* = 0$, where multiplying by the inverse A gives that

$$\boldsymbol{\mu} = \boldsymbol{\mu}_*. \tag{A.12}$$

This tells us that the mean learned via ℓ_2-robust maximum likelihood estimation is precisely the true mean of the distribution.

Now, in the same way, we set out to find $\boldsymbol{\Sigma}$ by solving the relevant system of equations:

$$\boldsymbol{\Sigma}_*^{-1} = \boldsymbol{\Sigma}^{-1}(M+I)^2. \tag{A.13}$$

Now, we make use of the Woodbury Matrix Identity in order to write $(I + M)$ as

$$I + (\lambda\boldsymbol{\Sigma} - I)^{-1} = I + \left(-I - \left(\frac{1}{\lambda}\boldsymbol{\Sigma}^{-1} - I\right)^{-1}\right) = -\left(\frac{1}{\lambda}\boldsymbol{\Sigma}^{-1} - I\right)^{-1}.$$

Thus, we can revisit (A.13) as follows:

$$\begin{aligned}
\boldsymbol{\Sigma}_*^{-1} &= \boldsymbol{\Sigma}^{-1}\left(\frac{1}{\lambda}\boldsymbol{\Sigma}^{-1} - I\right)^{-2} \\
\frac{1}{\lambda^2}\boldsymbol{\Sigma}_*^{-1}\boldsymbol{\Sigma}^{-2} - \left(\frac{2}{\lambda}\boldsymbol{\Sigma}_*^{-1} + I\right)\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_*^{-1} &= 0 \\
\frac{1}{\lambda^2}\boldsymbol{\Sigma}_*^{-1} - \left(\frac{2}{\lambda}\boldsymbol{\Sigma}_*^{-1} + I\right)\boldsymbol{\Sigma} + \boldsymbol{\Sigma}_*^{-1}\boldsymbol{\Sigma}^2 &= 0
\end{aligned}$$

We now apply the quadratic formula to get an implicit expression for $\boldsymbol{\Sigma}$ (implicit since technically λ depends on $\boldsymbol{\Sigma}$):

$$\begin{aligned}
\boldsymbol{\Sigma} &= \left(\frac{2}{\lambda}\boldsymbol{\Sigma}_*^{-1} + I \pm \sqrt{\frac{4}{\lambda}\boldsymbol{\Sigma}_*^{-1} + I}\right)\frac{1}{2}\boldsymbol{\Sigma}_* \\
&= \frac{1}{\lambda}I + \frac{1}{2}\boldsymbol{\Sigma}_* + \sqrt{\frac{1}{\lambda}\boldsymbol{\Sigma}_* + \frac{1}{4}\boldsymbol{\Sigma}_*^2}. \qquad \text{(A.14)}
\end{aligned}$$

This concludes the proof. □

Bounding λ

We now attempt to characterize the shape of λ as a function of ε. First, we use the fact that $\mathbb{E}[\|Xv\|^2] = \mathrm{tr}(X^2)$ for standard normally-drawn v. Thus, λ is set such that $\mathrm{tr}(\boldsymbol{\Sigma}_* M^2) = \varepsilon$, i.e:

$$\sum_{i=0} \frac{\boldsymbol{\Sigma}_{ii}^*}{(\lambda\boldsymbol{\Sigma}_{ii} - 1)^2} = \varepsilon \qquad \text{(A.15)}$$

Now, consider ε^2 as a function of λ. Observe that for $\lambda \geq \frac{1}{\sigma_{min}(\boldsymbol{\Sigma})}$, we have that M must be positive semi-definite, and thus ε^2 decays smoothly from ∞ (at $\lambda = \frac{1}{\sigma_{min}}$) to zero (at $\lambda = \infty$). Similarly, for $\lambda \leq \frac{1}{\sigma_{max}(\boldsymbol{\Sigma})}$, ε decays smoothly as λ *decreases*. Note, however, that such values of λ would necessarily make M *negative semi-definite*, which would actually *help* the log-likelihood. Thus, we can exclude this case; in particular, for the remainder of the proofs, we can assume $\lambda \geq \frac{1}{\sigma_{max}(\boldsymbol{\Sigma})}$.

Also observe that the zeros of ε in terms of λ are only at $\lambda = \pm\infty$. Using this, we can show that there exists some ε_0 for which, for all $\varepsilon < \varepsilon_0$, the only corresponding possible valid value of λ is where $\lambda \geq \frac{1}{\sigma_{min}}$. This idea is formalized in the following

Lemma.

Lemma 8. *For every $\mathbf{\Sigma}_*$, there exists some $\varepsilon_0 > 0$ for which, for all $\varepsilon \in [0, \varepsilon_0)$ the only admissible value of λ is such that $\lambda \geq \frac{1}{\sigma_{min}(\mathbf{\Sigma})}$, and thus such that M is positive semi-definite.*

Proof. We prove the existence of such an ε_0 by lower bounding ε (in terms of λ) for any finite $\lambda > 0$ that does not make M PSD. Providing such a lower bound shows that for small enough ε (in particular, less than this lower bound), the only corresponding values of λ are as desired in the statement[3].

In particular, if M is not PSD, then there must exist at least one index k such that $\lambda\mathbf{\Sigma}_{kk} < 1$, and thus $(\lambda\mathbf{\Sigma}_{kk} - 1)^2 \leq 1$ for all $\lambda > 0$. We can thus lower bound (A.15) as:

$$\varepsilon = \sum_{i=0} \frac{\mathbf{\Sigma}^*_{ii}}{(\lambda\mathbf{\Sigma}_{ii} - 1)^2} \geq \frac{\mathbf{\Sigma}^*_{kk}}{(\lambda\mathbf{\Sigma}_{kk} - 1)^2} \geq \mathbf{\Sigma}^*_{kk} \geq \sigma_{min}(\mathbf{\Sigma}^*) > 0 \tag{A.16}$$

By contradiction, it follows that for any $\varepsilon < \sigma_{min}(\mathbf{\Sigma}_*)^2$, the only admissible λ is such that M is PSD, i.e. according to the statement of the Lemma. □

In the regime $\varepsilon \in [0, \varepsilon_0)$, note that λ is inversely proportional to ε (i.e. as ε grows, λ decreases). This allows us to get a qualitative view of (A.14): as the allowed perturbation value increases, the robust covariance $\mathbf{\Sigma}$ resembles the identity matrix more and more, and thus assigns more and more variance on initially low-variance features. The $\sqrt{\mathbf{\Sigma}_*}$ term indicates that the robust model also adds uncertainty proportional to the square root of the initial variance—thus, low-variance features will have (relatively) more uncertainty in the robust case. Indeed, our main result actually follows as a (somewhat loose) formalization of this intuition.

Proof of main theorems

First, we give a proof of Theorem 2, providing lower and upper bounds on the learned robust covariance $\mathbf{\Sigma}$ in the regime $\varepsilon \in [0, \varepsilon_0)$.

[3]Since our only goal is existence, we lose many factors from the analysis that would give a tighter bound on ε_0.

Theorem 2 (Robustly Learned Parameters)**.** *Just as in the non-robust case,* $\boldsymbol{\mu}_r = \boldsymbol{\mu}^*$*, i.e. the true mean is learned. For the robust covariance* $\boldsymbol{\Sigma}_r$*, there exists an* $\varepsilon_0 > 0$*, such that for any* $\varepsilon \in [0, \varepsilon_0)$*,*

$$\boldsymbol{\Sigma}_r = \frac{1}{2}\boldsymbol{\Sigma}_* + \frac{1}{\lambda} \cdot \boldsymbol{I} + \sqrt{\frac{1}{\lambda} \cdot \boldsymbol{\Sigma}_* + \frac{1}{4}\boldsymbol{\Sigma}_*^2}, \quad \textit{where} \quad \Omega\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}+\varepsilon^{3/2}}\right) \leq \lambda \leq O\left(\frac{1+\varepsilon^{1/2}}{\varepsilon^{1/2}}\right).$$

Proof. We have already shown that $\boldsymbol{\mu} = \boldsymbol{\mu}_*$ in the robust case (c.f. (A.12)). We choose ε_0 to be as described, i.e. the largest ε for which the set $\{\lambda : \mathrm{tr}(\boldsymbol{\Sigma}_*^2 M) = \varepsilon, \lambda \geq 1/\sigma_{\max}(\boldsymbol{\Sigma})\}$ has only one element λ (which, as we argued, must not be less than $1/\sigma_{min}(\boldsymbol{\Sigma})$). We have argued that such an ε_0 must exist.

We prove the result by combining our early derivation (in particular, (A.13) and (A.14)) with upper and lower bound on λ, which we can compute based on properties of the trace operator. We begin by deriving a lower bound on λ. By linear algebraic manipulation (given in Appendix A.1.3), we get the following bound:

$$\lambda \geq \frac{d}{\mathrm{tr}(\boldsymbol{\Sigma})}\left(1 + \sqrt{\frac{d \cdot \sigma_{min}(\boldsymbol{\Sigma}_*)}{\varepsilon}}\right) \tag{A.17}$$

Now, we can use (A.13) in order to remove the dependency of λ on $\boldsymbol{\Sigma}$:

$$\begin{aligned}
\boldsymbol{\Sigma} &= \boldsymbol{\Sigma}_*(M+I)^2 \\
\mathrm{tr}(\boldsymbol{\Sigma}) &= \mathrm{tr}\left[(\boldsymbol{\Sigma}_*^{1/2}M + \boldsymbol{\Sigma}_*^{1/2})^2\right] \\
&\leq 2 \cdot \mathrm{tr}\left[(\boldsymbol{\Sigma}_*^{1/2}M)^2 + (\boldsymbol{\Sigma}_*^{1/2})^2\right] \\
&\leq 2 \cdot (\varepsilon + \mathrm{tr}(\boldsymbol{\Sigma}_*)).
\end{aligned}$$

Applying this to (A.17) yields:

$$\lambda \geq \frac{d/2}{\varepsilon + \mathrm{tr}(\boldsymbol{\Sigma}_*)}\left(1 + \sqrt{\frac{d \cdot \sigma_{min}(\boldsymbol{\Sigma}_*)}{\varepsilon}}\right).$$

Note that we can simplify this bound significantly by writing $\varepsilon = d \cdot \sigma_{min}(\boldsymbol{\Sigma}_*)\varepsilon' \leq \mathrm{tr}(\boldsymbol{\Sigma}_*)\varepsilon'$, which does not affect the result (beyond rescaling the valid regime $(0, \varepsilon_0)$),

and gives:

$$\lambda \geq \frac{d/2}{(1+\varepsilon')\mathrm{tr}(\mathbf{\Sigma}_*)}\left(1+\frac{1}{\sqrt{\varepsilon'}}\right) \geq \frac{d\cdot(1+\sqrt{\varepsilon'})}{2\sqrt{\varepsilon'}(1+\varepsilon')\mathrm{tr}(\mathbf{\Sigma}_*)}$$

Next, we follow a similar methodology (Appendix A.1.3) in order to upper bound λ:

$$\lambda \leq \frac{1}{\sigma_{min}(\mathbf{\Sigma})}\left(\sqrt{\frac{\|\mathbf{\Sigma}_*\|_F \cdot d}{\varepsilon}}+1\right).$$

Note that by (A.13) and positive semi-definiteness of M, it must be that $\sigma_{min}(\mathbf{\Sigma}) \geq \sigma_{min}(\mathbf{\Sigma}_*)$. Thus, we can simplify the previous expression, also substituting $\varepsilon = d\cdot\sigma_{min}(\mathbf{\Sigma}_*)\varepsilon'$:

$$\lambda \leq \frac{1}{\sigma_{min}(\mathbf{\Sigma}_*)}\left(\sqrt{\frac{\|\mathbf{\Sigma}_*\|_F}{\sigma_{min}(\mathbf{\Sigma}_*)\varepsilon'}}+1\right) = \frac{\|\mathbf{\Sigma}_*\|_F + \sqrt{\varepsilon\cdot\sigma_{min}(\mathbf{\Sigma}_*)}}{\sigma_{min}(\mathbf{\Sigma}_*)^{3/2}\sqrt{\varepsilon}}$$

These bounds can be straightforwardly combined with Lemma 7, which concludes the proof. □

Using this theorem, we can now show Theorem 3:

Theorem 3 (Gradient alignment)**.** *Let $f(x)$ and $f_r(x)$ be monotonic classifiers based on the linear separator induced by standard and ℓ_2-robust maximum likelihood classification, respectively. The maximum angle formed between the gradient of the classifier (wrt input) and the vector connecting the classes can be smaller for the robust model:*

$$\min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f_r(x)\rangle}{\|\boldsymbol{\mu}\|\cdot\|\nabla_x f_r(x)\|} > \min_{\boldsymbol{\mu}} \frac{\langle \boldsymbol{\mu}, \nabla_x f(x)\rangle}{\|\boldsymbol{\mu}\|\cdot\|\nabla_x f(x)\|}.$$

Proof. To prove this, we make use of the following Lemmas:

Lemma 9. *For two positive definite matrices A and B with $\kappa(A) > \kappa(B)$, we have that $\kappa(A+B) \leq \max\{\kappa(A), \kappa(B)\}$.*

Proof. We proceed by contradiction:

$$\begin{aligned}
\kappa(A+B) &= \frac{\lambda_{max}(A)+\lambda_{max}(B)}{\lambda_{min}(A)+\lambda_{min}(B)} \\
\kappa(A) &= \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \\
\kappa(A) &\geq \kappa(A+B) \\
\leftrightarrow \lambda_{max}(A)\left(\lambda_{min}(A)+\lambda_{min}(B)\right) &\geq \lambda_{min}(A)\left(\lambda_{max}(A)+\lambda_{max}(B)\right) \\
\leftrightarrow \lambda_{max}(A)\lambda_{min}(B) &\geq \lambda_{min}(A)\lambda_{max}(B) \\
\leftrightarrow \frac{\lambda_{max}(A)}{\lambda_{min}(A)} &\geq \frac{\lambda_{min}(A)}{\lambda_{max}(B)},
\end{aligned}$$

which is false by assumption. This concludes the proof. □

Lemma 10 (Straightforward)**.** *For a positive definite matrix A and $k > 0$, we have that*

$$\kappa(A + k \cdot I) < \kappa(A) \qquad \kappa(A + k \cdot \sqrt{A}) \leq \kappa(A).$$

Lemma 11 (Angle induced by positive definite matrix; folklore)**.** [4] *For a positive definite matrix $A \succ 0$ with condition number κ, we have that*

$$\min_x \frac{x^\top A x}{\|Ax\|_2 \cdot \|x\|_2} = \frac{2\sqrt{\kappa}}{1+\kappa}. \tag{A.18}$$

These two results can be combined to prove the theorem. First, we show that

[4] A proof can be found in `https://bit.ly/2L6jdAT`

$\kappa(\mathbf{\Sigma}) \leq \kappa(\mathbf{\Sigma}_*)$:

$$
\begin{aligned}
\kappa(\mathbf{\Sigma}) &= \kappa\left(\frac{1}{\lambda}I + \frac{1}{2}\mathbf{\Sigma}_* + \sqrt{\frac{1}{\lambda}\mathbf{\Sigma}_* + \frac{1}{4}\mathbf{\Sigma}_*^2}\right) \\
&< \max\left\{\kappa\left(\frac{1}{\lambda}I + \frac{1}{2}\mathbf{\Sigma}_*\right), \kappa\left(\sqrt{\frac{1}{\lambda}\mathbf{\Sigma}_* + \frac{1}{4}\mathbf{\Sigma}_*^2}\right)\right\} \\
&< \max\left\{\kappa\left(\mathbf{\Sigma}_*\right), \sqrt{\kappa\left(\frac{1}{\lambda}\mathbf{\Sigma}_* + \frac{1}{4}\mathbf{\Sigma}_*^2\right)}\right\} \\
&= \max\left\{\kappa\left(\mathbf{\Sigma}_*\right), \sqrt{\kappa\left(\frac{2}{\lambda}\sqrt{\frac{1}{4}\mathbf{\Sigma}_*^2} + \frac{1}{4}\mathbf{\Sigma}_*^2\right)}\right\} \\
&\leq \kappa\left(\mathbf{\Sigma}_*\right).
\end{aligned}
$$

Finally, note that (A.18) is a strictly decreasing function in κ, and as such, we have shown the theorem. □

Bounds for λ

Lower bound.

$$
\begin{aligned}
\varepsilon &= \operatorname{tr}(\mathbf{\Sigma}_* M^2) \\
&\geq \sigma_{min}(\mathbf{\Sigma}_*) \cdot \operatorname{tr}(M^2) && \text{by the definition of } \operatorname{tr}(\cdot) \\
&\geq \frac{\sigma_{min}(\mathbf{\Sigma}_*)}{d} \cdot \operatorname{tr}(M)^2 && \text{by Cauchy-Schwarz} \\
&\geq \frac{\sigma_{min}(\mathbf{\Sigma}_*)}{d} \cdot \left[\operatorname{tr}\left((\lambda\mathbf{\Sigma} - \boldsymbol{I})^{-1}\right)\right]^2 && \text{Expanding } M \text{ (A.9)} \\
&\geq \frac{\sigma_{min}(\mathbf{\Sigma}_*)}{d} \cdot \left[\operatorname{tr}\left(\lambda\mathbf{\Sigma} - \boldsymbol{I}\right)^{-1} \cdot d^2\right]^2 && \text{AM-HM inequality} \\
&\geq d^3 \cdot \sigma_{min}(\mathbf{\Sigma}_*) \cdot \left[\lambda \cdot \operatorname{tr}(\mathbf{\Sigma}) - d\right]^{-2} \\
\left[\lambda \cdot \operatorname{tr}(\mathbf{\Sigma}) - d\right]^2 &\geq \frac{d^3 \cdot \sigma_{min}(\mathbf{\Sigma}_*)}{\varepsilon} \\
\lambda \cdot \operatorname{tr}(\mathbf{\Sigma}) - d &\geq \frac{d^{3/2} \cdot \sqrt{\sigma_{min}(\mathbf{\Sigma}_*)}}{\sqrt{\varepsilon}} && \text{since } M \text{ is PSD} \\
\lambda &\geq \frac{d}{\operatorname{tr}(\mathbf{\Sigma})}\left(1 + \sqrt{\frac{d \cdot \sigma_{min}(\mathbf{\Sigma}_*)}{\varepsilon}}\right)
\end{aligned}
$$

Upper bound

$$
\begin{aligned}
\varepsilon &= \mathrm{tr}(\mathbf{\Sigma}_* M^2) \\
&\leq \|\mathbf{\Sigma}_*\|_F \cdot d \cdot \sigma_{max}(M)^2 \\
&\leq \|\mathbf{\Sigma}_*\|_F \cdot d \cdot \sigma_{min}(M)^{-2} \\
\lambda \cdot \sigma_{min}(\mathbf{\Sigma}) - 1 &\leq \sqrt{\frac{\|\mathbf{\Sigma}_*\|_F \cdot d}{\varepsilon}} \\
\lambda &\leq \frac{1}{\sigma_{min}(\mathbf{\Sigma})} \left(\sqrt{\frac{\|\mathbf{\Sigma}_*\|_F \cdot d}{\varepsilon}} + 1 \right).
\end{aligned}
$$

A.2 Experimental Setup for Robust Representations

A.2.1 Datasets

For our experimental analysis, we use the CIFAR-10 [Kri09] and ImageNet [Rus+15] datasets. Since obtaining a robust classifier for the full ImageNet dataset is known to be a challenging and computationally expensive problem, we also conduct experiments on a "restricted" version if the ImageNet dataset with 9 super-classes shown in Table A.1. For image translation we use the Horse ↔ Zebra, Apple ↔ Orange, and Summer ↔ Winter datasets [Zhu+17].

A.2.2 Models

We use the ResNet-50 architecture for our baseline standard and adversarially trained classifiers on CIFAR-10 and restricted ImageNet. For each model, we grid search over three learning rates (0.1, 0.01, 0.05), two batch sizes (128, 256) including/not including a learning rate drop (a single order of magnitude) and data augmentation. We use the standard training parameters for the remaining parameters. The hyperparameters used for each model are given in Table A.2. Additionally, it has been noted in prior work that among standard classifiers, VGG networks [SZ15] tend to have better-behaved

Class	Corresponding ImageNet Classes
"Dog"	151 to 268
"Cat"	281 to 285
"Frog"	30 to 32
"Turtle"	33 to 37
"Bird"	80 to 100
"Primate"	365 to 382
"Fish"	389 to 397
"Crab"	118 to 121
"Insect"	300 to 319

Table A.1: Classes used in the Restricted ImageNet model. The class ranges are inclusive.

representations and feature visualizations [Mor+18]. Thus, we also compare the visualization experiments against standard VGG16 networks.

Dataset	LR	Batch Size	LR Drop	Data Aug.	Momentum	Weight Decay
$\widehat{\mathcal{D}}_R$ (CIFAR)	0.1	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_R$ (Restricted ImageNet)	0.01	128	No	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{NR}$ (CIFAR)	0.1	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{rand}$ (CIFAR)	0.01	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{rand}$ (Restricted ImageNet)	0.01	256	No	No	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{det}$ (CIFAR)	0.1	128	Yes	No	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{det}$ (Restricted ImageNet)	0.05	256	No	No	0.9	$5 \cdot 10^{-4}$

Table A.2: Hyperparameters for the models trained in the main paper. All hyperparameters were obtained through a grid search.

A.2.3 Adversarial training

To obtain robust classifiers, we employ the adversarial training methodology proposed in [Mad+18]. Specifically, we train against a projected gradient descent (PGD) adversary constrained in ℓ_2-norm starting from the original image. Following Madry et al. [Mad+18] we normalize the gradient at each step of PGD to ensure that we move a fixed distance in ℓ_2-norm per step. Unless otherwise specified, we use the values of ε provided in Table A.3 to train/evaluate our models. We used 7 steps of PGD with a step size of $\varepsilon/5$.

Adversary	**CIFAR-10**	**Restricted Imagenet**
ℓ_2	0.5	3

Table A.3: Value of ε used for ℓ_2 adversarial training/evaluation of each dataset.

A.2.4 Constructing a Robust Dataset

In Section 2.5.1, we describe a procedure to construct a dataset that contains features relevant only to a given (standard/robust) model. To do so, we optimize the training objective in (2.8). Unless otherwise specified, we initialize x_r as a different randomly chosen sample from the training set. (For the sake of completeness, we also try initializing with a Gaussian noise instead as shown in Table A.10.) We then perform normalized gradient descent (ℓ_2-norm of gradient is fixed to be constant at each step). At each step we clip the input x_r to in the $[0, 1]$ range so as to ensure that it is a valid image. Details on the optimization procedure are shown in Table A.4. We provide the pseudocode for the construction in Figure A-1.

GetRobustDataset(D)

1. $C_R \leftarrow$ AdversarialTraining(D)
   $g_R \leftarrow$ mapping learned by C_R from the input to the representation layer
2. $D_R \leftarrow \{\}$
3. For $(x, y) \in D$
   $x' \sim D$
   $x_R \leftarrow \arg\min_{z \in [0,1]^d} \|g_R(z) - g_R(x)\|_2$
   # Solved using ℓ_2-PGD starting from x'
   $D_R \leftarrow D_R \bigcup \{(x_R, y)\}$
4. Return D_R

Figure A-1: Algorithm to construct a "robust" dataset, by restricting to features used by a robust model.

	CIFAR-10	**Restricted Imagenet**
step size	0.1	1
iterations	1000	2000

Table A.4: Parameters used for optimization procedure to construct dataset in Section 2.5.1.

A.2.5 Non-robust features suffice for standard classification

To construct the dataset as described in Section 2.5.2, we use the standard projected gradient descent (PGD) procedure described in [Mad+18] to construct an adversarial example for a given input from the dataset (2.9). Perturbations are constrained in ℓ_2-norm while each PGD step is normalized to a fixed step size. The details for our PGD setup are described in Table A.5. We provide pseudocode in Figure A-2.

GetNonRobustDataset(D, ε)

1. $D_{NR} \leftarrow \{\}$
2. $C \leftarrow$ StandardTraining(D)
3. For $(x, y) \in D$
   $t \overset{\text{uar}}{\sim} [C]$ # or $t \leftarrow (y+1) \mod C$
   $x_{NR} \leftarrow \min_{||x'-x|| \leq \varepsilon} L_C(x', t)$
   # Solved using ℓ_2 PGD
   $D_{NR} \leftarrow D_{NR} \bigcup \{(x_{NR}, t)\}$
4. Return D_{NR}

Figure A-2: Algorithm to construct a dataset where input-label association is based entirely on non-robust features.

Attack Parameters	CIFAR-10	Restricted Imagenet
ε	0.5	3
step size	0.1	0.1
iterations	100	100

Table A.5: Projected gradient descent parameters used to construct constrained adversarial examples in Section 2.5.2.

A.2.6 Image interpolations

A natural consequence of the "natural invertibility" property of robust representations is the ability to synthesize natural interpolations between any two inputs $x_1, x_2 \in \mathbb{R}^n$.

In particular, given two images x_1 and x_2, we define the λ-*interpolate* between them as

$$x_\lambda = \min_x \| (\lambda \cdot R(x_1) + (1-\lambda) \cdot R(x_2)) - R(x) \|_2. \tag{A.19}$$

where, for a given λ, we find x_λ by solving (A.19) with projected gradient descent. Intuitively, this corresponds to linearly interpolating between the points in representation space and then finding a point in image space that has a similar representation. To construct a length-$(T+1)$ interpolation, we choose $\lambda = \{0, \frac{1}{T}, \frac{2}{T}, \ldots 1\}$. The resulting interpolations, shown in Figure A-3, demonstrate that the λ-interpolates of robust representations correspond to a meaningful feature interpolation between images. (For standard models constructing meaningful interpolations is impossible due to the brittleness identified in Section 3.2—see Appendix A.3.13 for details.)

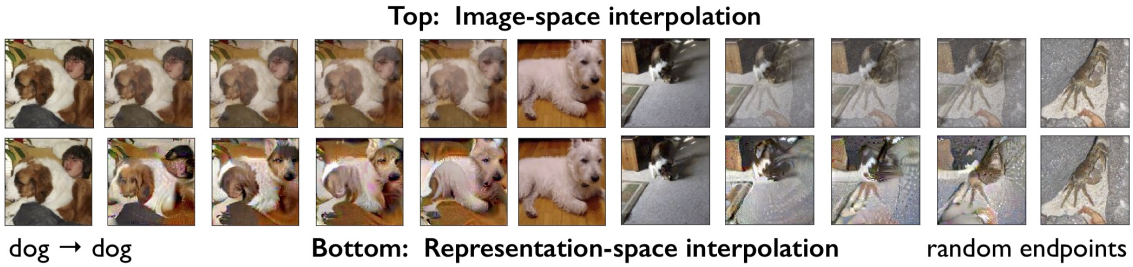


Figure A-3: Image interpolation using robust representations compared to their image-space counterparts. The former appear perceptually plausible while the latter exhibit ghosting artifacts. For pairs of images from the Restricted ImageNet test set, we solve (A.19) for λ varying between zero and one, i.e., we match linear interpolates in representation space. Additional interpolations appear in Appendix A.3.14 Figure A-25. We demonstrate the ineffectiveness of interpolation with standard representations in Appendix A.3.14 Figure A-26.

Relation to other interpolation methods. We emphasize that linearly interpolating in robust representation space works for *any* two images. This generality is in contrast to interpolations induced by GANs (e.g. [RMC16; BDS19]), which can only interpolate between images generated by the generator. (Reconstructions of out-of-range images tend to be decipherable but rather different from the originals [Bau+19b].) It is worth noting that even for models with analytically invertible representations, interpolating in representation space does not yield semantic interpolations [JSO18].

A.2.7 Parameters used in studies of robust/standard representations

Finding representation-feature correspondence

Dataset	ε	# steps	Step size
Restricted ImageNet/ImageNet	1000	200	1

Inverting representations and interpolations

Dataset	ε	# steps	Step size
Restricted ImageNet/ImageNet	1000	10000	1

A.2.8 Targeted Attacks in Figure 3-13

Dataset	ε	# steps	Step size
restricted ImageNet	300	500	1

A.2.9 Image-to-image translation

Dataset	ε	# steps	Step size
ImageNet	60	80	1
Horse $\leftrightarrow$ Zebra	60	80	0.5
Apple $\leftrightarrow$ Orange	60	80	0.5
Summer $\leftrightarrow$ Winter	60	80	0.5

A.2.10 Generation

In order to compute the class conditional Gaussians for high resolution ($224\times224\times3$) images we downsample the images by a factor of 4 and upsample the resulting seed images with nearest neighbor interpolation.

Dataset	ε	# steps	Step size
CIFAR-10	30	60	0.5
restricted ImageNet	40	60	1
ImageNet	40	60	1

Inception Score

Inception score is computed based on 50k class-balanced samples from each dataset using code provided in `https://github.com/ajbrock/BigGAN-PyTorch`.

A.2.11 Inpainting

To create a corrupted image, we select a patch of a given size at a random location in the image. We reset all pixel values in the patch to be the average pixel value over the entire image (per channel).

Dataset	patch size	ε	# steps	Step size
restricted ImageNet	60	21	0.1	720

A.2.12 Super-resolution

Dataset	$\uparrow$ factor	ε	# steps	Step size
CIFAR-10	7	15	1	50
restricted ImageNet	8	8	1	40

A.3 Omitted Experiments and Figures

A.3.1 Spatial Robustness

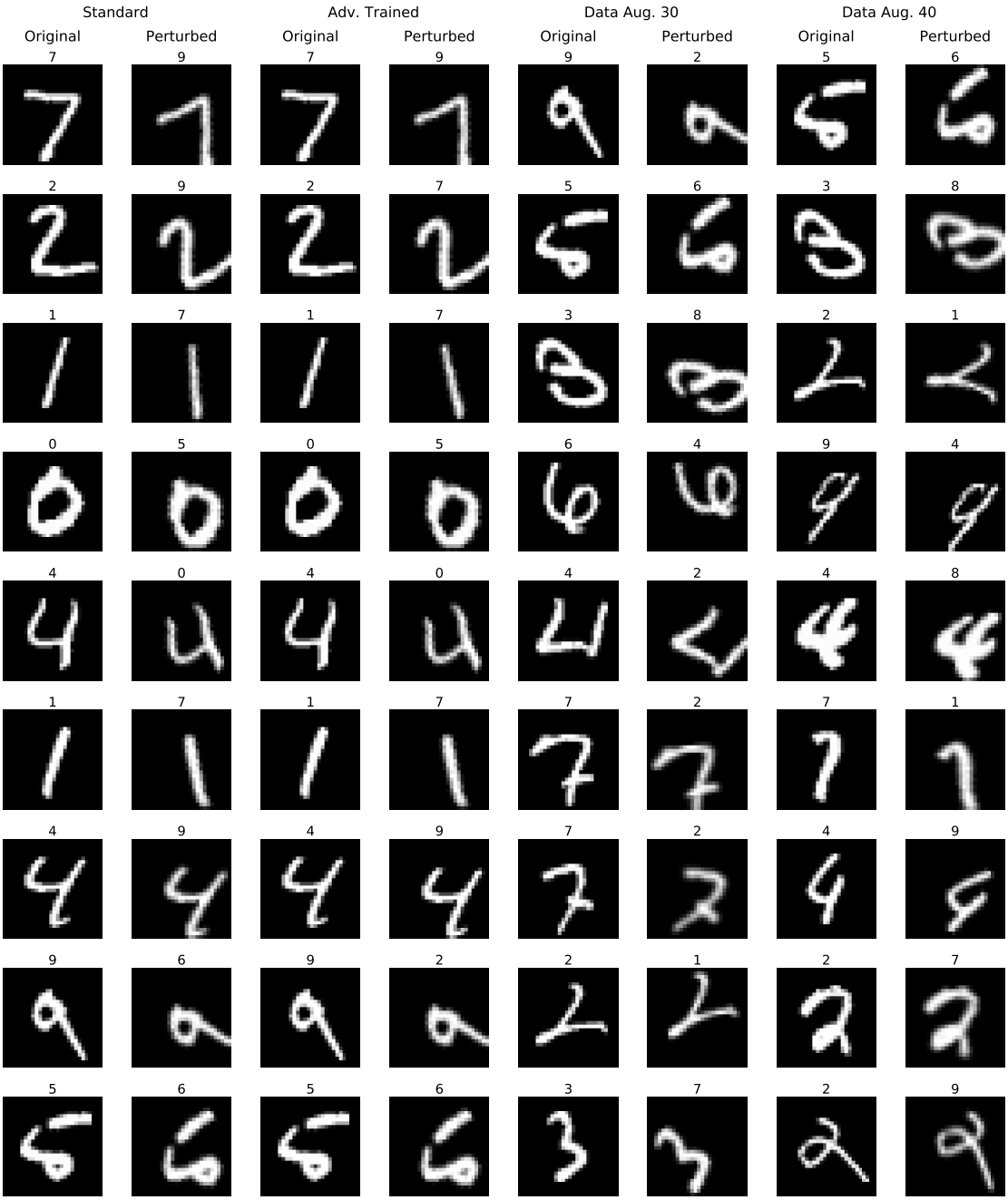


Figure A-4: MNIST. Successful adversarial examples for the models studied in Section 2.10. Rotations are restricted to be within 30° of the original image and translations up to 3 pixels per direction (image size 28×28). Each example is visualized along with its predicted label in the original and perturbed versions.

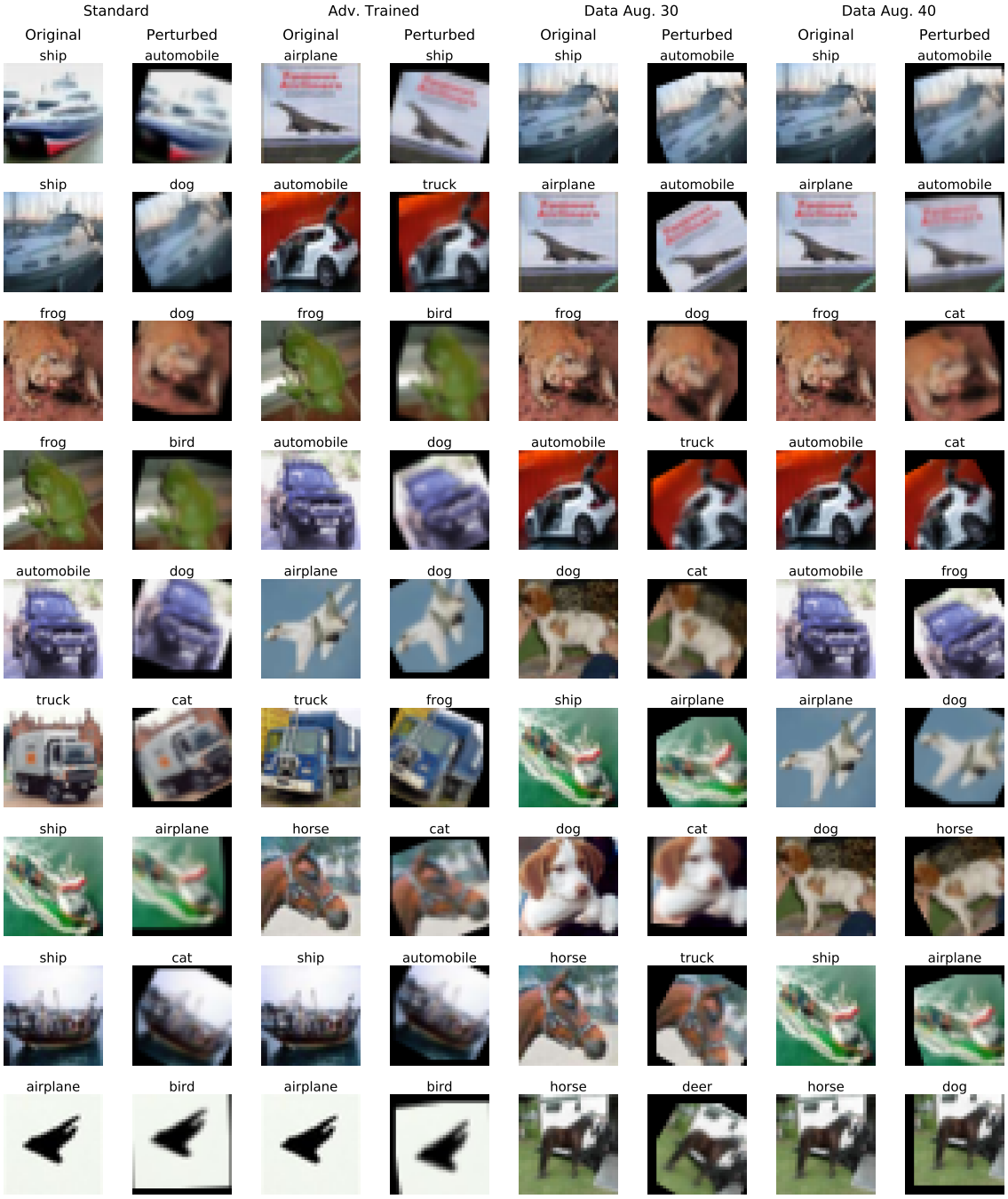


Figure A-5: CIFAR10. Successful adversarial examples for the models studied in Section 2.10. Rotations are restricted to be within 30° of the original and translations up to 3 pixels per directions (image size 32×32). Each example is visualized along with its predicted label in the original and perturbed version.

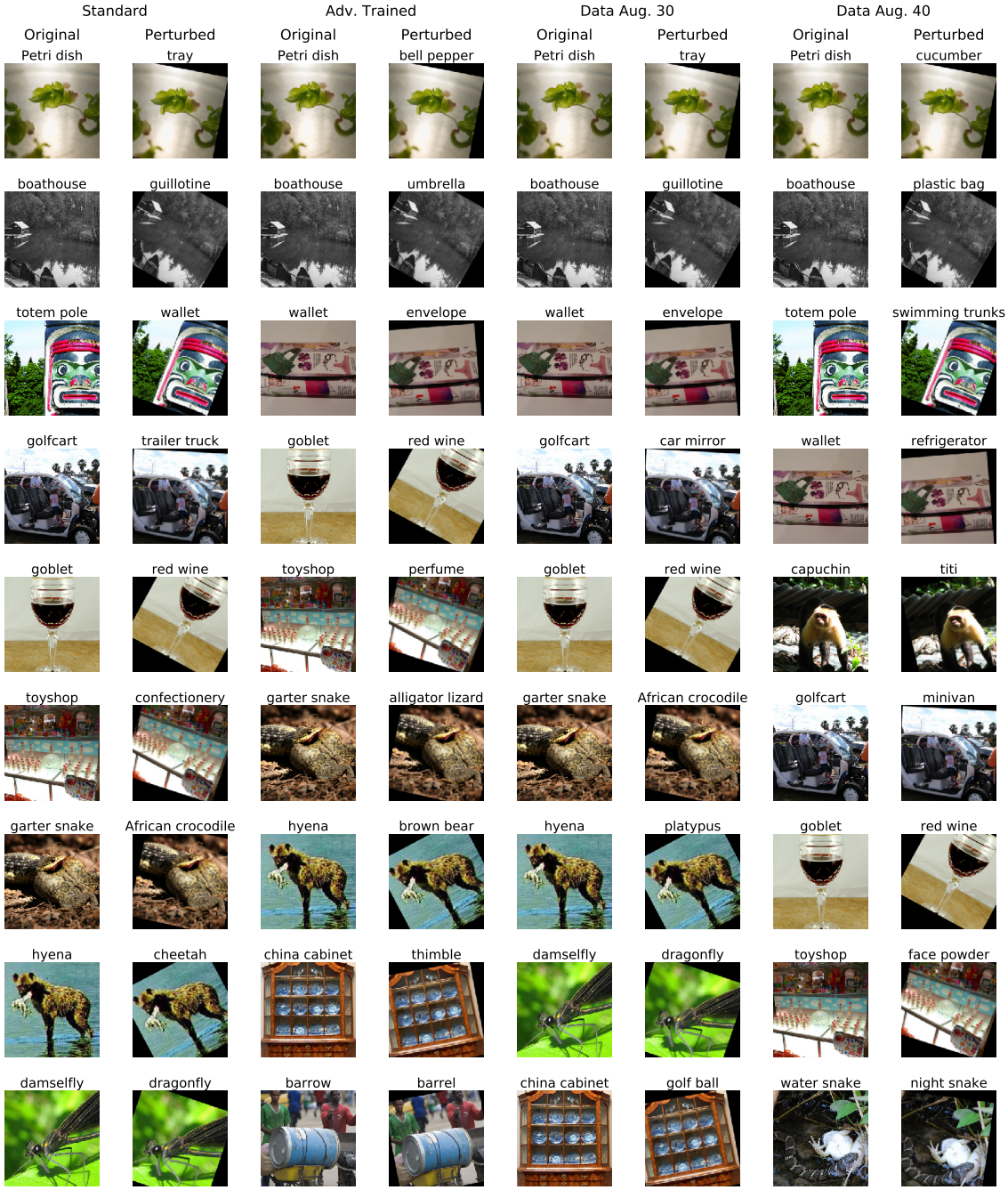


Figure A-6: ImageNet. Successful adversarial examples for the models studied in Section 2.10. Rotations are restricted to be within 30° of the original and translations up to 24 pixels per directions (image size 299×299). Each example is visualized along with its predicted label in the original and perturbed version.

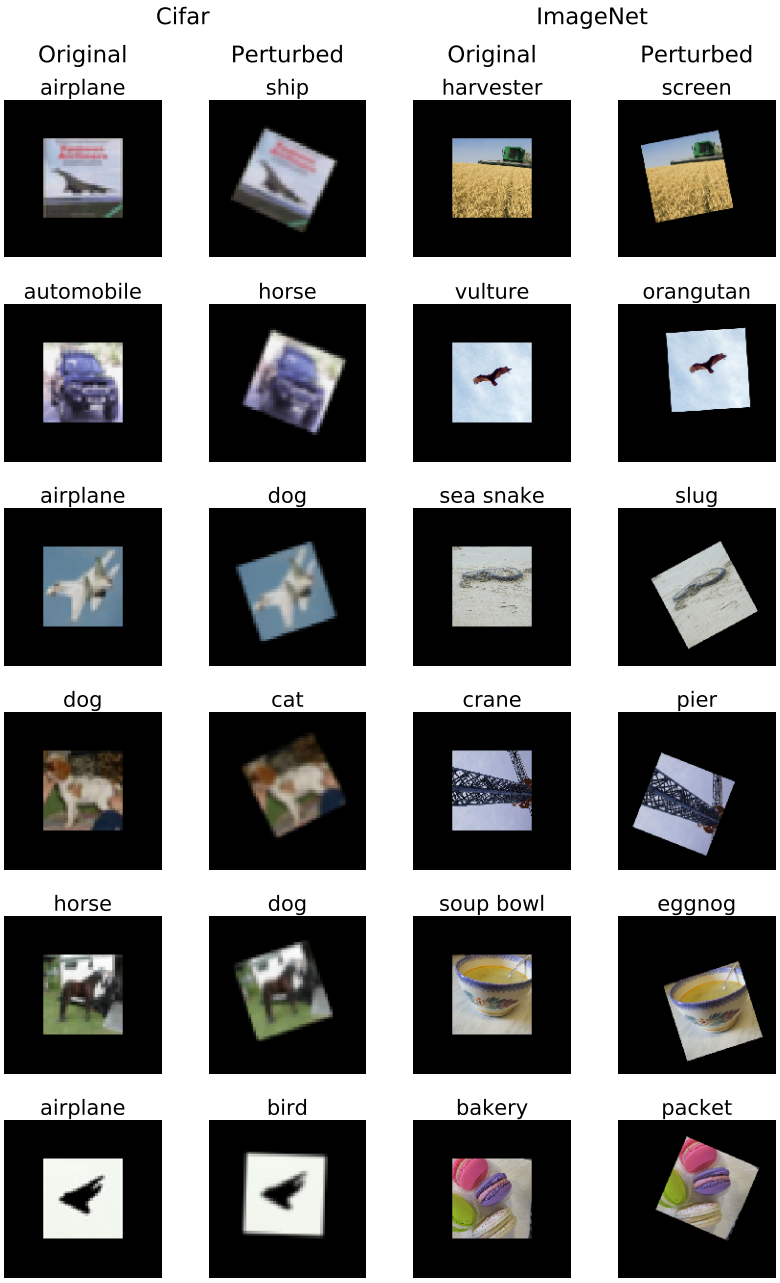


Figure A-7: Sample adversarial transformations for the "black-canvas" setting for the standard models on CIFAR10 and ImageNet.

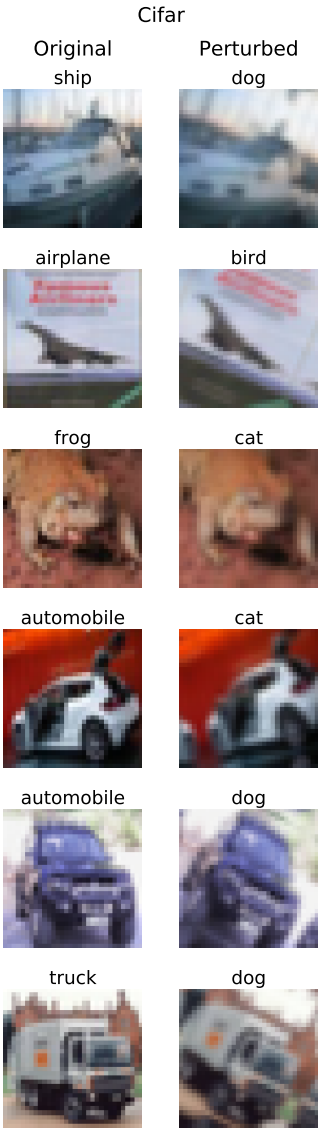


Figure A-8: Sample adversarial transformations for the reflection padding setting for the standard models on CIFAR10.

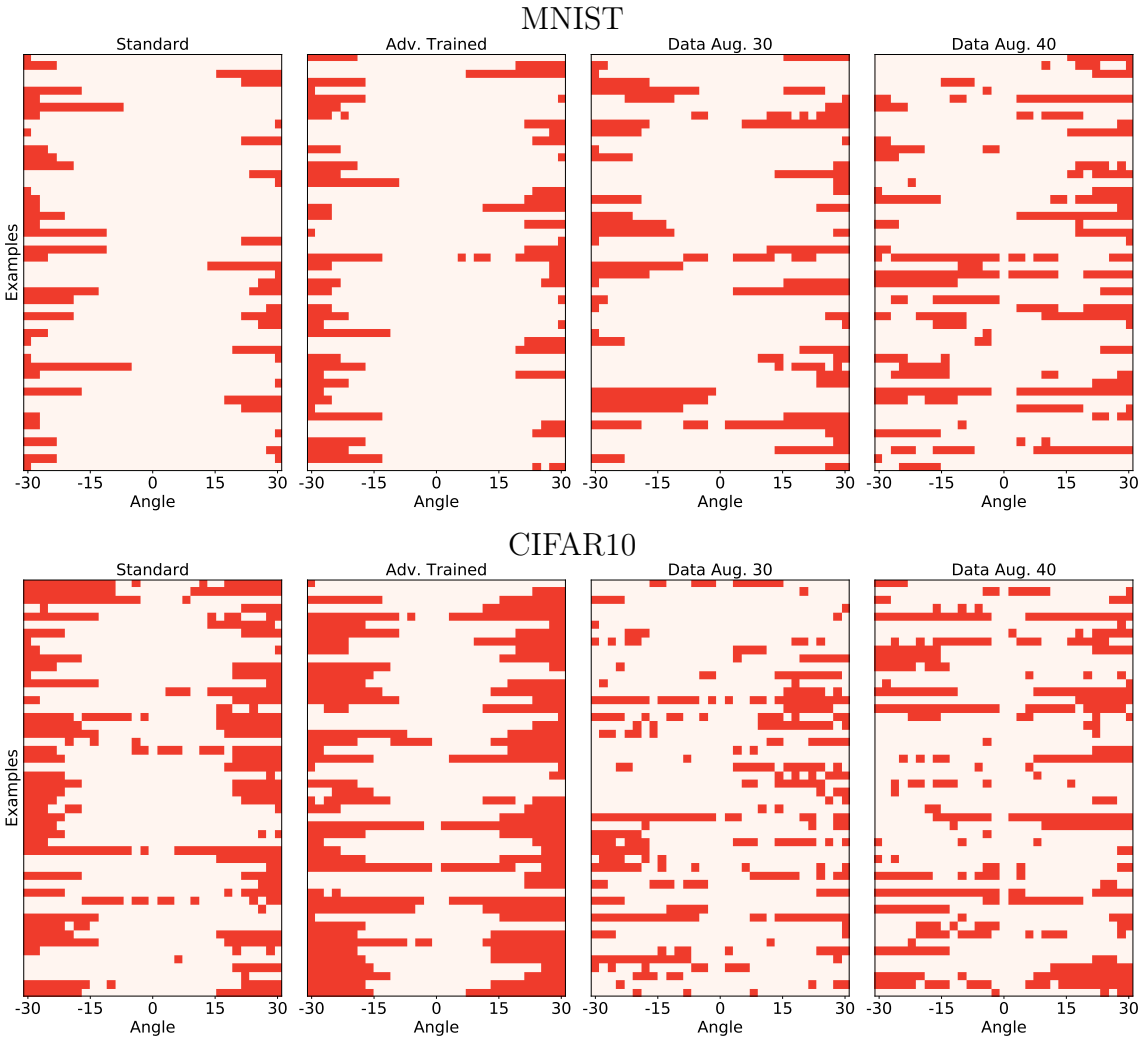


Figure A-9: Visualizing which angles fool the classifier for 50 random examples on CIFAR and MNIST. For each dataset and model, we visualize one example per row. Red corresponds to *misclassification* of the images. We observe that the angles fooling the models form a highly non-convex set.

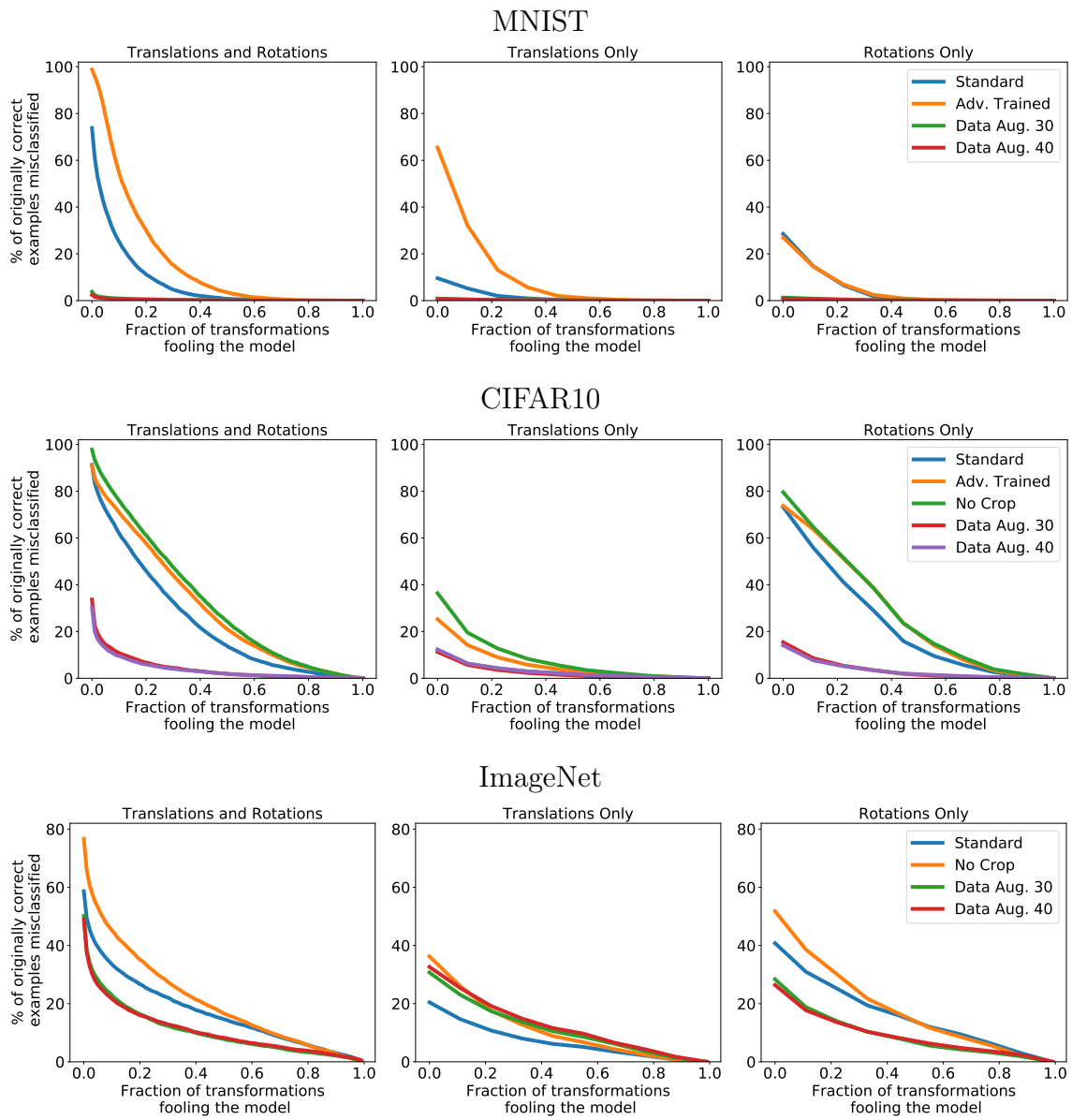


Figure A-10: Cumulative Density Function plots. For each fraction of grid points p, we plot the percentage of correctly classified test set examples that are fooled by at least p of the grid points. For instance, we can see from the first plot, MNIST Translations and Rotations, that approximately 10% of the correctly classified natural examples are misclassified under 1/5 of the grid points transformations.

	Model	Natural	Worst-of-10	FO	Grid
MNIST	Standard	99.31%	73.32%	79.84%	**26.02**%
	ℓ_∞-Adversarially Trained	98.65%	51.18%	81.23%	**1.20**%
	Aug. 30 ($\pm$3px, $\pm$30°)	99.53%	98.33%	98.78%	**95.79**%
	Aug. 40 ($\pm$4px, $\pm$40°)	99.34%	98.49%	98.74%	**96.95**%
CIFAR10	Standard	92.62%	20.13%	62.69%	**2.80**%
	No Crop	90.34%	15.04%	52.27%	**1.86**%
	ℓ_∞-Adversarially Trained	80.21%	19.38%	33.24%	**6.02**%
	Aug. 30 ($\pm$3px, $\pm$30°)	90.02%	79.92%	85.92%	**58.92**%
	Aug. 40 ($\pm$4px, $\pm$40°)	88.83%	80.47%	85.48%	**61.69**%
ImageNet	Standard	75.96%	47.83%	63.12%	**31.42**%
	No Crop	70.81%	35.52%	55.93%	**16.52**%
	Aug. 30 ($\pm$24px, $\pm$30°)	65.96%	50.62%	66.05%	**32.90**%
	Aug. 40 ($\pm$32px, $\pm$40°)	66.19%	51.11%	66.14%	**33.86**%

Table A.6: Comparison of attack methods across datasets and models.

		Natural	Random	Worst-of-10	Grid	Trans. Grid	Rot. Grid
CIFAR10	Standard	91.81%	70.23%	25.51%	**6.55**%	83.38%	12.44%
	No Crop	89.70%	52.86%	14.14%	**1.17**%	47.94%	9.46%
	Aug. 30 ($\pm$3px, $\pm$30°)	91.45%	90.82%	80.53%	**63.64**%	82.28%	76.32%
	Aug. 40 ($\pm$4px, $\pm$40°)	91.24%	91.00%	81.81%	**66.64**%	81.75%	78.57%
ImageNet	Standard	73.60%	46.59%	29.51%	**15.38**%	28.03%	23.81%
	No Crop	66.28%	38.70%	14.17%	**3.43**%	8.87%	10.97%
	Aug. 30 ($\pm$24px, $\pm$30°)	64.60%	67.75%	47.32%	**28.51**%	45.33%	39.33%
	Aug. 40 ($\pm$32px, $\pm$40°)	49.20%	57.69%	38.36%	**22.10**%	32.84%	32.95%

Table A.7: Evaluation of a subset of Table 2.2 in the "black-canvas" setting (images are zero-padded to avoid cropping due to rotations and translations). The models are trained on padded images.

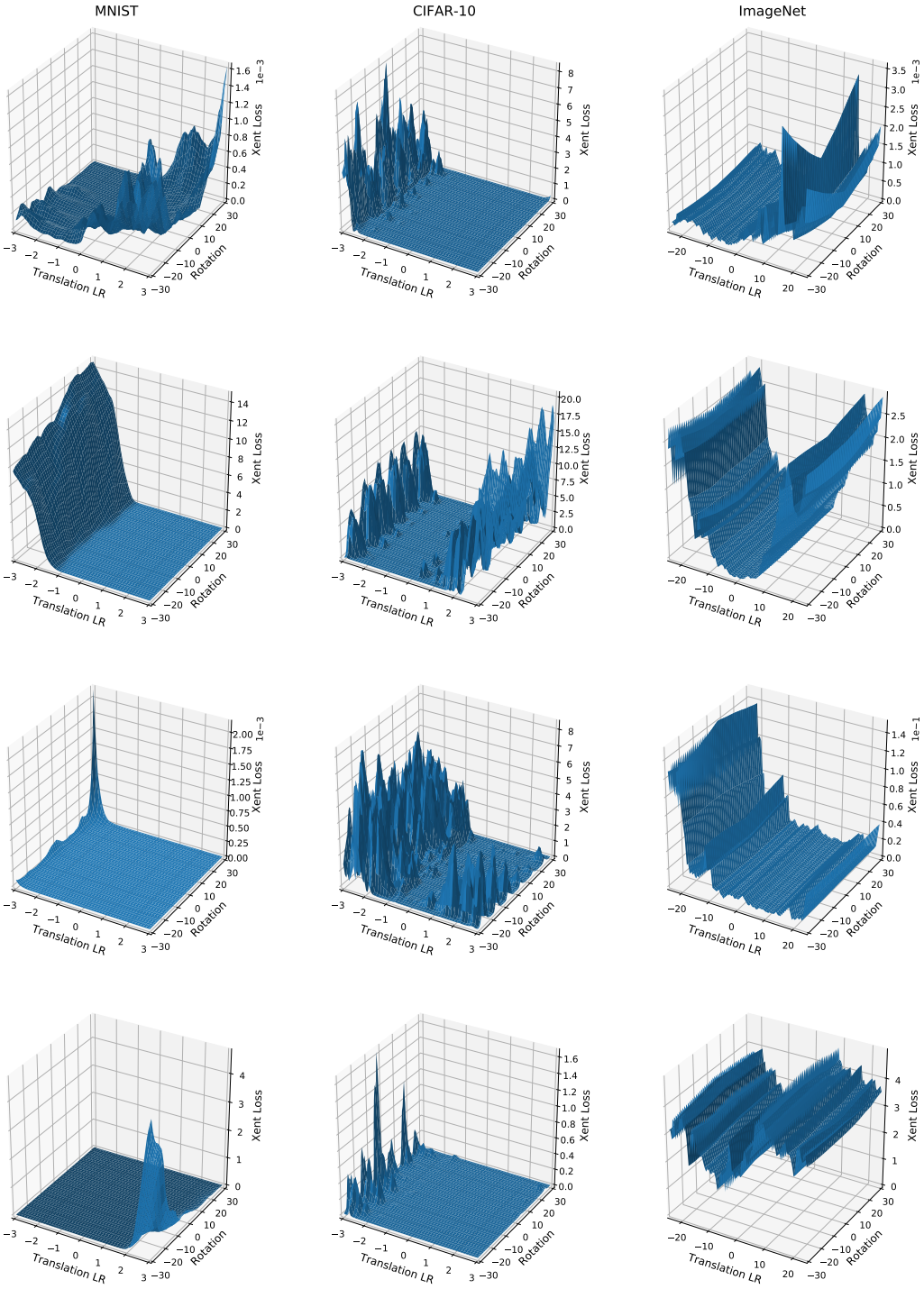


Figure A-11: Loss landscape of 4 random examples for each dataset when performing left-right translations and rotations. Translations and rotations are restricted to 10% of the image pixels and 30° respectively. We observe that the landscape is significantly non-concave, making rendering FO methods for adversarial example generation powerless.

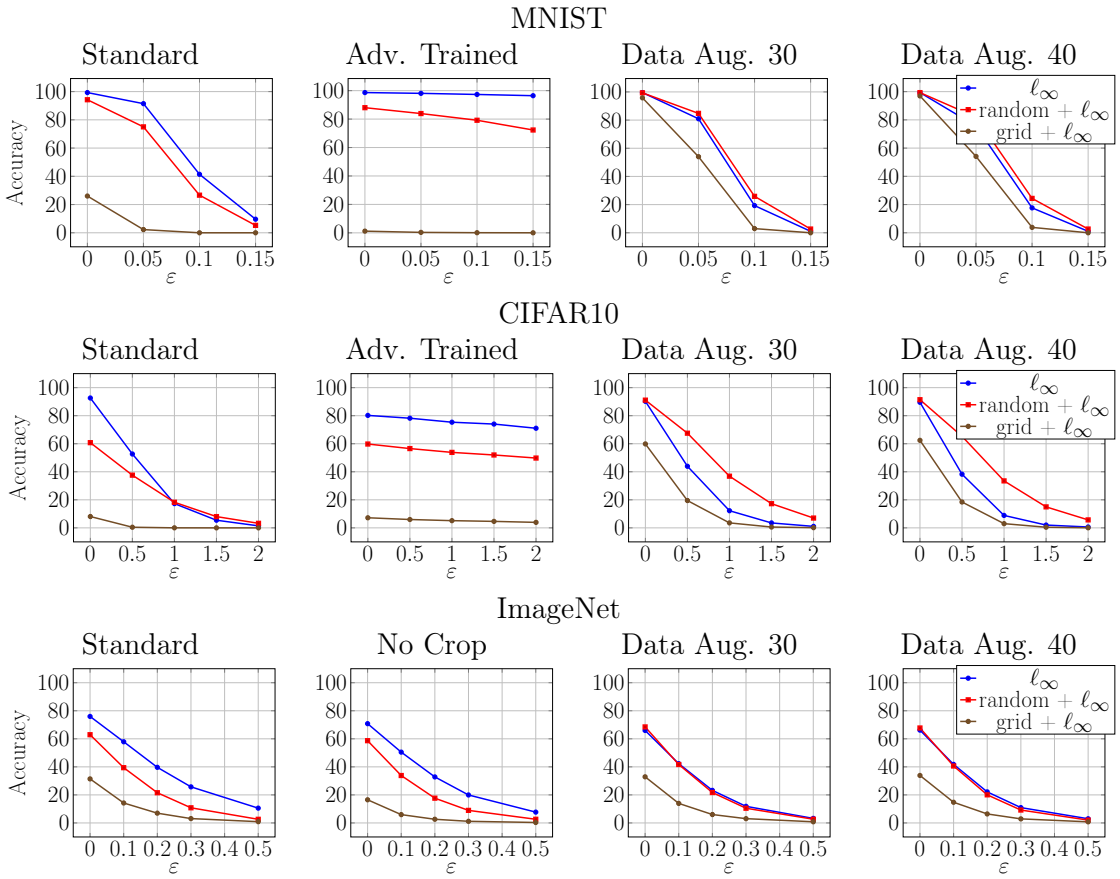


Figure A-12: Accuracy of different classifiers against ℓ_∞-bounded adversaries with various values of ε and spatial transformations. For each value of ε, we perform PGD to find the most adversarial ℓ_∞-bounded perturbation. Additionally, we combine PGD with random rotations and translations and with a grid search over rotations and translations in order to find the transformation that combines with PGD in the most adversarial way.

A.3.2 Mirror Padding

In the experiments of Section 2.10, we filled the remaining pixels of rotated and translated images with black (also known as zero or constant padding). This is the standard approach used when performing random cropping for data augmentation purposes. We briefly examined the effect of mirror padding, that is replacing empty pixels by reflecting the image around the border[5]. The results are shown in Table A.9.

[5] `https://www.tensorflow.org/api_docs/python/tf/pad`

	Model	Natural Acc. Stand.	Natural Acc. Vote	Grid Acc. Stand.	Grid Acc. Vote
MNIST	Standard	99.31%	98.71%	26.02%	18.80%
	Aug 30.	**99.53**%	99.41%	95.79%	95.32%
	Aug 40.	99.34%	99.25%	96.95%	97.65%
	W-10 (30)	99.48%	99.40%	97.32%	96.95%
	W-10 (40)	99.42%	99.41%	97.88%	**98.47**%
CIFAR10	Standard	92.62%	80.37%	2.82%	7.85%
	Aug 30.	90.02%	92.70%	58.90%	69.65%
	Aug 40.	88.83%	92.50%	61.69%	76.54%
	W-10 (30)	91.34%	93.38%	69.17%	77.33%
	W-10 (40)	91.00%	**93.40**%	71.15%	**81.52**%
ImageNet	Standard	75.96%	73.19%	31.42%	40.21%
	Aug 30.	65.96%	72.44%	32.90%	44.46%
	Aug 40.	66.19%	71.46%	33.86%	46.98%
	W-10 (30)	**76.14**%	74.92%	52.76%	**56.45**%
	W-10 (40)	74.64%	73.38%	50.23%	56.23%

Table A.8: Majority Defense. Accuracy of different models on the natural evaluation set and against a combined rotation and translation adversary using aggregation of multiple random transformations.

We observed that training with one padding method and evaluating using the other resulted in a significant drop in accuracy. Training using one of these methods randomly for each example resulted in a model which roughly matched the best-case of the two individual cases.

A.3.3 Detailed evaluation of models trained on "robust" dataset

In Section 2.5.1, we generate a "robust" training set by restricting the dataset to only contain features relevant to a robust model (robust dataset) or a standard model (non-robust dataset). This is performed by choosing either a random input from the training set or random noise[6] and then performing the optimization procedure

[6]We use 10k steps to construct the dataset from noise, instead to using 1k steps done when the input is a different training set image (cf. Table A.4).

	Natural	Random (Zero)	Random (Mirror)	Grid Search (Zero)	Grid Search (Mirror)
Standard Nat	92.62%	60.76%	66.42%	8.08%	5.37%
Standard Adv	80.21%	59.79%	67.12%	7.20%	12.89%
Aug. A, Zero	90.25%	91.09%	87.67%	59.87%	40.55%
Aug. B, Zero	89.55%	91.40%	87.94%	62.42%	42.37%
Aug. A, Mirror	92.25%	88.43%	91.05%	41.46%	53.95%
Aug. B, Mirror	92.03%	88.58%	91.34%	45.44%	57.97%
Aug. A, Both	91.80%	90.98%	91.28%	56.95%	52.60%
Aug. B, Both	91.57%	91.87%	91.11%	60.46%	56.13%

Table A.9: CIFAR10: The effect of using reflection or zero padding when training a model. The experimental setup matches that of Section 2.10. Zero padding refers to filling the empty pixels caused by translations and rotations with black. Mirror padding corresponds to using a reflection of the images. "Both" refers to training using both methods and alternating randomly between them for each training example.

described in (2.8). The performance of these classifiers along with various baselines is shown in Table A.10. We observe that while the robust dataset constructed from noise resembles the original, the corresponding non-robust does not (Figure A-13). This also leads to suboptimal performance of classifiers trained on this dataset (only 46% standard accuracy) potentially due to a distributional shift.

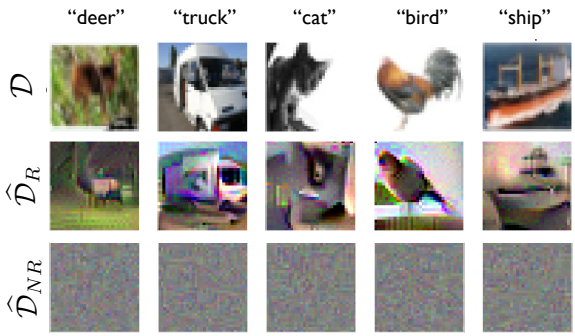


Figure A-13: Robust and non-robust datasets for CIFAR-10 when the process starts from noise (as opposed to random images as in Figure 2-3a).

Model	Accuracy	Robust Accuracy $\varepsilon = 0.25$	$\varepsilon = 0.5$
Standard Training	95.25 %	4.49%	0.0%
Robust Training	90.83%	82.48%	70.90%
Trained on non-robust dataset (from images)	**87.68%**	0.82%	0.0%
Trained on non-robust dataset (from noise)	**45.60%**	1.50%	0.0%
Trained on robust dataset (from images)	85.40%	**48.20 %**	21.85%
Trained on robust dataset (from noise)	84.10%	**48.27 %**	29.40%

Table A.10: Standard and robust classification performance on the CIFAR-10 test set of: an (i) ERM classifier; (ii) ERM classifier trained on a dataset obtained by distilling features relevant to ERM classifier in (i); (iii) adversarially trained classifier ($\varepsilon = 0.5$); (iv) ERM classifier trained on dataset obtained by distilling features used by robust classifier in (iii). Simply restricting the set of available features during ERM to features used by a standard model yields non-trivial robust accuracy.

A.3.4 Adversarial evaluation

To verify the robustness of our classifiers trained on the 'robust" dataset, we evaluate them with strong attacks [Car+19]. In particular, we try up to 2500 steps of projected gradient descent (PGD), increasing steps until the accuracy plateaus, and also try the CW-ℓ_2 loss function [CW17b] with 1000 steps. For each attack we search over step size. We find that over all attacks and step sizes, the accuracy of the model does not drop by more than 2%, and plateaus at 48.27% for both PGD and CW-ℓ_2 (the value given in Figure 2-3). We show a plot of accuracy in terms of the number of PGD steps used in Figure A-14.

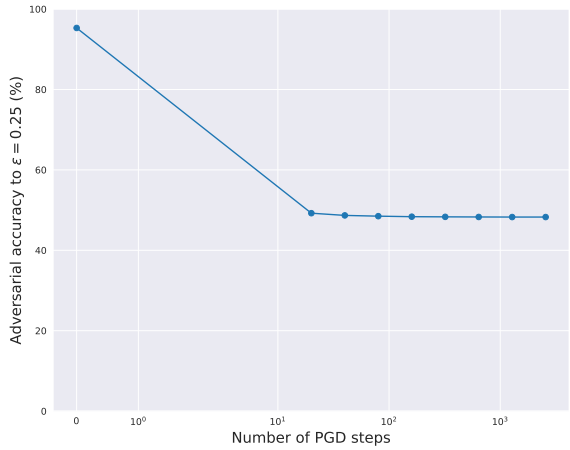


Figure A-14: Robust accuracy as a function of the number of PGD steps used to generate the attack. The accuracy plateaus at 48.27%.

A.3.5 Performance of "robust" training and test set

In Section 2.5.1, we observe that an ERM classifier trained on a "robust" training dataset $\widehat{\mathcal{D}}_R$ (obtained by restricting features to those relevant to a robust model) attains non-trivial robustness (cf. Figure 2-2 and Table A.10). In Table A.11, we evaluate the adversarial accuracy of the model on the corresponding robust training set (the samples which the classifier was trained on) and test set (unseen samples from $\widehat{\mathcal{D}}_R$, based on the test set). We find that the drop in robustness comes from a combination of generalization gap (the robustness on the $\widehat{\mathcal{D}}_R$ test set is worse than it

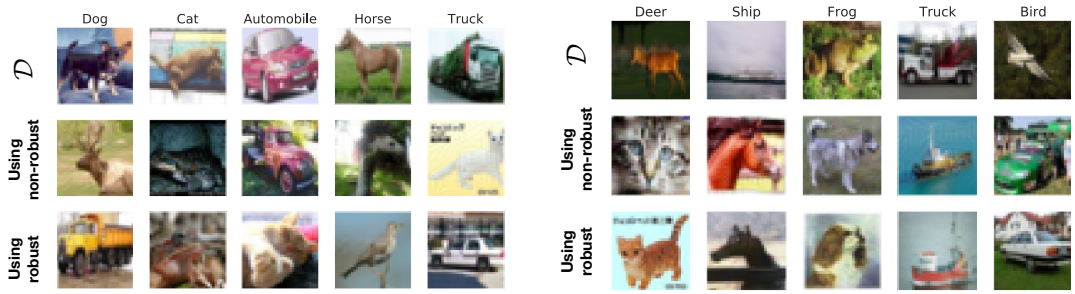
is on the robust training set) and distributional shift (the model performs better on the robust test set consisting of unseen samples from $\widehat{\mathcal{D}}_R$ than on the standard test set containing unseen samples from $\mathcal{D}$).

Dataset	Robust Accuracy
Robust training set	77.33%
Robust test set	62.49%
Standard test set	48.27%

Table A.11: Performance of model trained on the *robust dataset* on the robust training and test sets as well as the standard CIFAR-10 test set. We observe that the drop in robust accuracy stems from a combination of generalization gap and distributional shift. The adversary is constrained to $\varepsilon = 0.25$ in ℓ_2-norm.

A.3.6 Classification based on non-robust features

Figure A-15 shows sample images from $\mathcal{D}$, $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$ constructed using a standard (non-robust) ERM classifier, and an adversarially trained (robust) classifier.



(a) $\widehat{\mathcal{D}}_{rand}$ (b) $\widehat{\mathcal{D}}_{det}$

Figure A-15: Random samples from datasets where the input-label correlation is entirely based on non-robust features. Samples are generated by performing small adversarial perturbations using either random ($\widehat{\mathcal{D}}_{rand}$) or deterministic ($\widehat{\mathcal{D}}_{det}$) label-target mappings for every sample in the training set. Each image shows: *top*: original; *middle*: adversarial perturbations using a standard ERM-trained classifier; *bottom*: adversarial perturbations using a robust classifier (adversarially trained against $\varepsilon = 0.5$).

In Table A.12, we repeat the experiments in Table 2.1 based on datasets constructed using a robust model. Note that using a robust model to generate the $\widehat{\mathcal{D}}_{det}$ and $\widehat{\mathcal{D}}_{rand}$

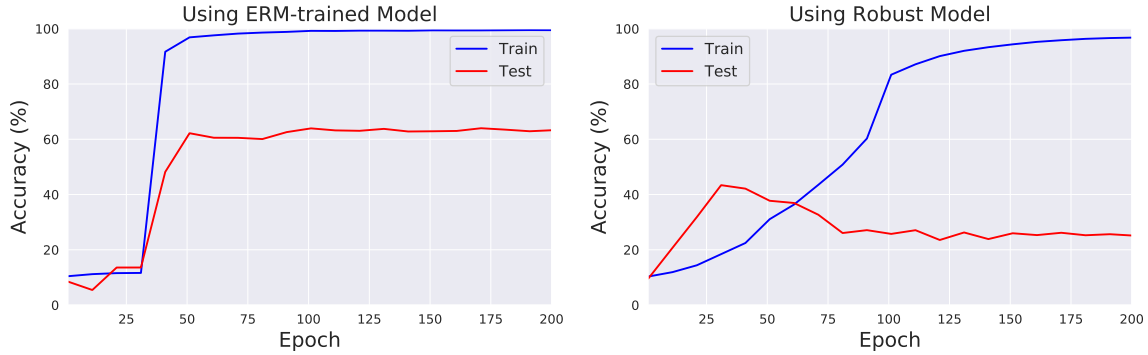
datasets will not result in non-robust features that are strongly predictive of t (since the prediction of the classifier will not change). Thus, training a model on these datasets leads to poor accuracy on the standard test set from $\mathcal{D}$.

Observe from Figure A-16 that models trained on datasets derived from the robust model show a decline in test accuracy as training progresses. In Table A.12, the accuracy numbers reported correspond to the *last* iteration, and not the *best* performance. This is because we have no way to cross-validate in a meaningful way as the validation set itself comes from $\widehat{\mathcal{D}}_{rand}$ or $\widehat{\mathcal{D}}_{det}$, and not from the true data distribution D. Thus, validation accuracy will not be predictive of the true test accuracy, and thus will not help determine when to early stop.

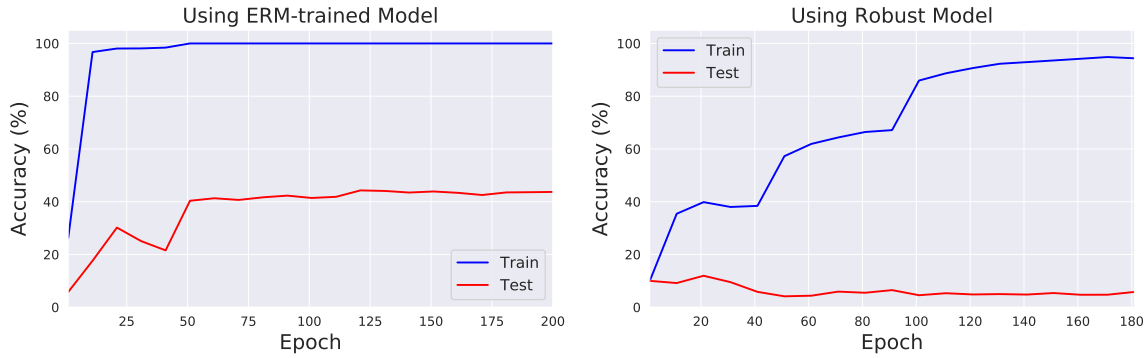
Model used to construct dataset	**Dataset used in training**		
	$\mathcal{D}$	$\widehat{\mathcal{D}}_{rand}$	$\widehat{\mathcal{D}}_{det}$
Robust	95.3%	25.2 %	5.8%
Standard	95.3%	63.3 %	43.7%

Table A.12: Repeating the experiments of Table 2.1 using a robust model to construct the datasets $\mathcal{D}$, $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$. Results in Table 2.1 are reiterated for comparison.

A.3.7 Accuracy curves



(a) Trained using $\widehat{\mathcal{D}}_{rand}$ training set



(b) Trained using $\widehat{\mathcal{D}}_{det}$ training set

Figure A-16: Test accuracy on $\mathcal{D}$ of standard classifiers trained on datasets where input-label correlation is based solely on non-robust features as in Section 2.5.2. The datasets are constructed using either a non-robust/standard model (*left column*) or a robust model (*right column*). The labels used are either random ($\widehat{\mathcal{D}}_{rand}$; *top row*) or correspond to a deterministic permutation ($\widehat{\mathcal{D}}_{det}$; *bottom row*).

A.3.8 Performance of ERM classifiers on relabeled test set

In Table A.13), we evaluate the performance of classifiers trained on $\widehat{\mathcal{D}}_{det}$ on both the original test set drawn from $\mathcal{D}$, and the test set relabelled using $t(y) = (y + 1) \mod C$. Observe that the classifier trained on $\widehat{\mathcal{D}}_{det}$ constructed using a robust model actually ends up learning permuted labels based on robust features (indicated by high test accuracy on the relabelled test set).

Model used to construct training dataset for $\widehat{\mathcal{D}}_{det}$	**Dataset used in testing**	
	$\mathcal{D}$	relabelled-$\mathcal{D}$
Standard	43.7%	16.2%
Robust	5.8%	65.5%

Table A.13: Performance of classifiers trained using $\widehat{\mathcal{D}}_{det}$ training set constructed using either standard or robust models. The classifiers are evaluated both on the standard test set from $\mathcal{D}$ and the test set relabeled using $t(y) = (y + 1) \mod C$. We observe that using a robust model for the construction results in a model that largely predicts the permutation of labels, indicating that the dataset does not have strongly predictive non-robust features.

A.3.9 Generalization to CIFAR-10.1

Recht et al. [Rec+19] have constructed an unseen but distribution-shifted test set for CIFAR-10. They show that for many previously proposed models, accuracy on the CIFAR-10.1 test set can be predicted as a linear function of performance on the CIFAR-10 test set.

As a sanity check (and a safeguard against any potential adaptive overfitting to the test set via hyperparameters, historical test set reuse, etc.) we note that the classifiers trained on $\widehat{\mathcal{D}}_{det}$ and $\widehat{\mathcal{D}}_{rand}$ achieve 44% and 55% generalization on the CIFAR-10.1 test set, respectively. This demonstrates non-trivial generalization, and actually perform better than the linear fit would predict (given their accuracies on the CIFAR-10 test set).

A.3.10 Omitted Results for Restricted ImageNet

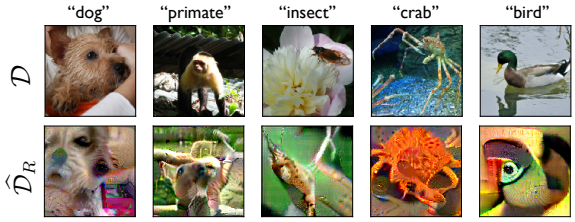


Figure A-17: Repeating the experiments shown in Figure 2-3 for the Restricted ImageNet dataset. Sample images from the resulting dataset.

Figure A-18: Repeating the experiments shown in Figure 2-3 for the Restricted ImageNet dataset. Standard and robust accuracy of models trained on these datasets.

A.3.11 Targeted Transferability

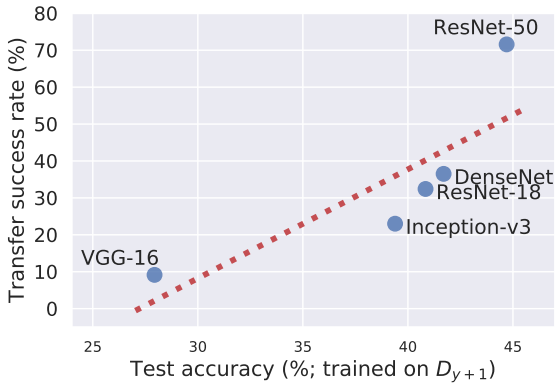


Figure A-19: Transfer rate of *targeted* adversarial examples (measured in terms of attack success rate, not just misclassification) from a ResNet-50 to different architectures alongside test set performance of these architecture when trained on the dataset generated in Section 2.5.2. Architectures more susceptible to transfer attacks also performed better on the standard test set supporting our hypothesis that adversarial transferability arises from utilizing similar *non-robust features*.

A.3.12 Robustness vs. Accuracy

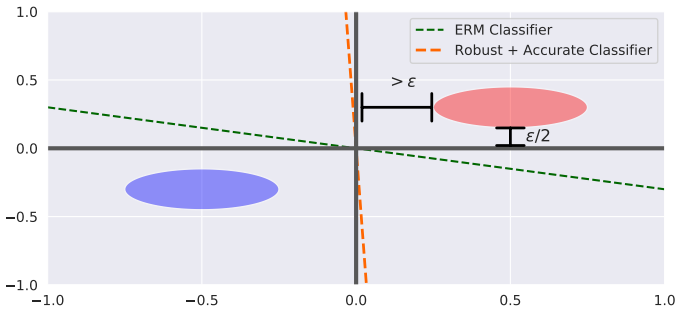
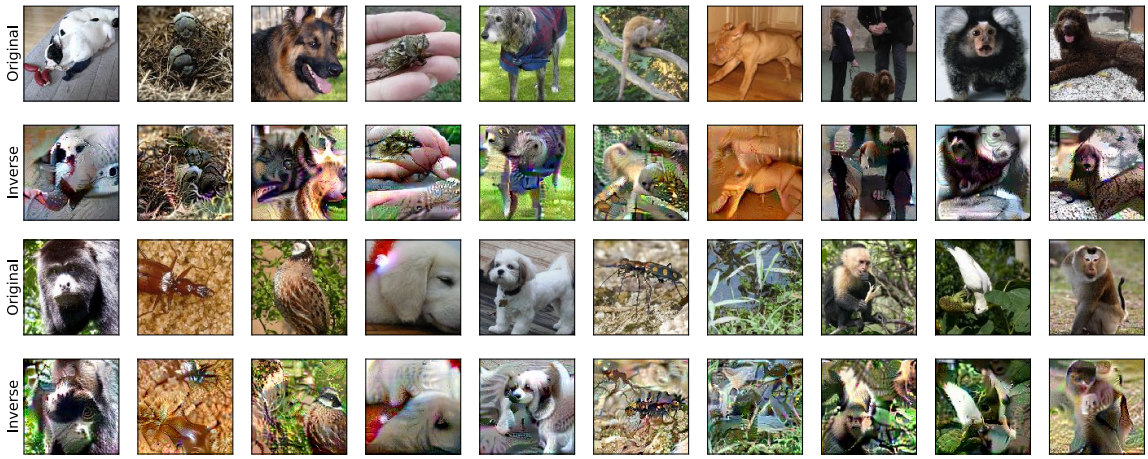


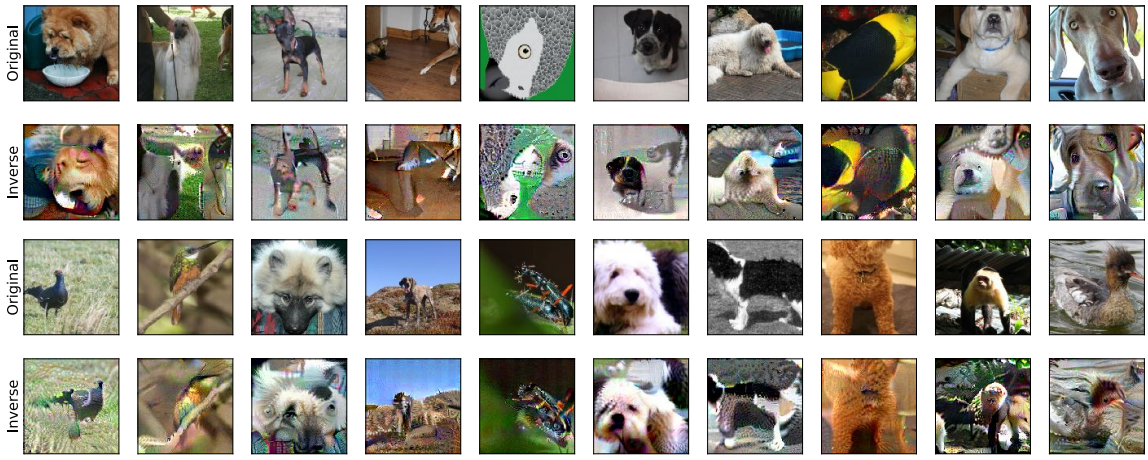
Figure A-20: An example where adversarial vulnerability can arise from ERM training on any standard loss function due to non-robust features (the green line shows the ERM-learned decision boundary). There exists, however, a classifier that is both perfectly robust *and* accurate, resulting from robust training, which forces the classifier to ignore the x_2 feature despite its predictiveness.

A.3.13 Inverting representations

Recovering test set images using robust representations



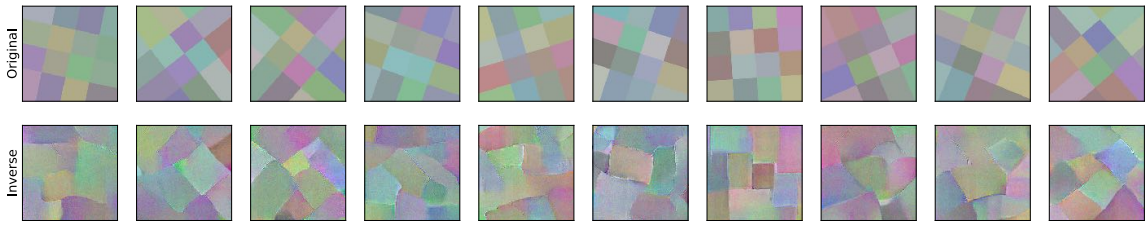
(a)



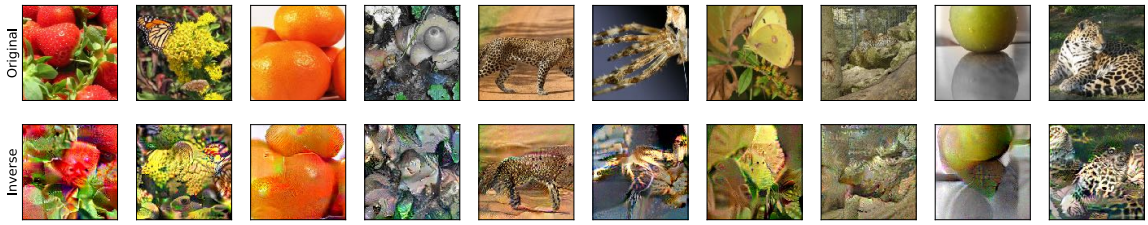
(b)

Figure A-21: Robust representations yield semantically meaningful inverses: *Original*: randomly chosen test set images from the Restricted ImageNet dataset; *Inverse*: images obtained by inverting the representation of the corresponding image in the top row by solving the optimization problem (3.1) starting from: (a) different test images and (b) Gaussian noise.

Recovering out-of-distribution inputs using robust representations



(a) Random kaleidoscope patterns.



(b) Samples from other ImageNet classes outside what the model is trained on.

Figure A-22: Robust representations yield semantically meaningful inverses: (*Original*): randomly chosen out-of-distribution inputs; (*Inverse*): images obtained by inverting the representation of the corresponding image in the top row by solving the optimization problem (3.1) starting from Gaussian noise.

Inverting standard representations

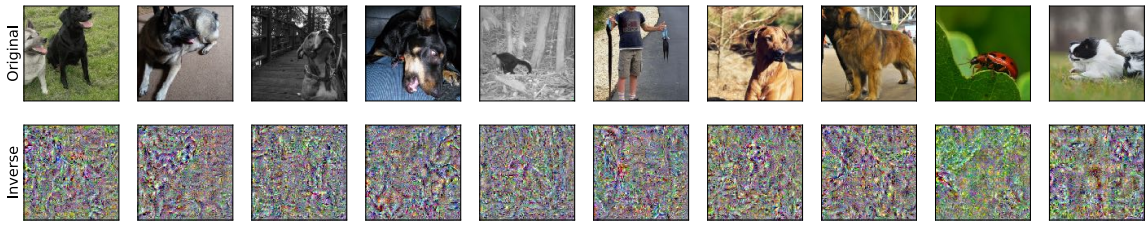


Figure A-23: Standard representations *do not* yield semantically meaningful inverses: (*Original*): randomly chosen test set images from the Restricted ImageNet dataset; (*Inverse*): images obtained by inverting the representation of the corresponding image in the top row by solving the optimization problem (3.1) starting from Gaussian noise.

Representation inversion on the ImageNet dataset

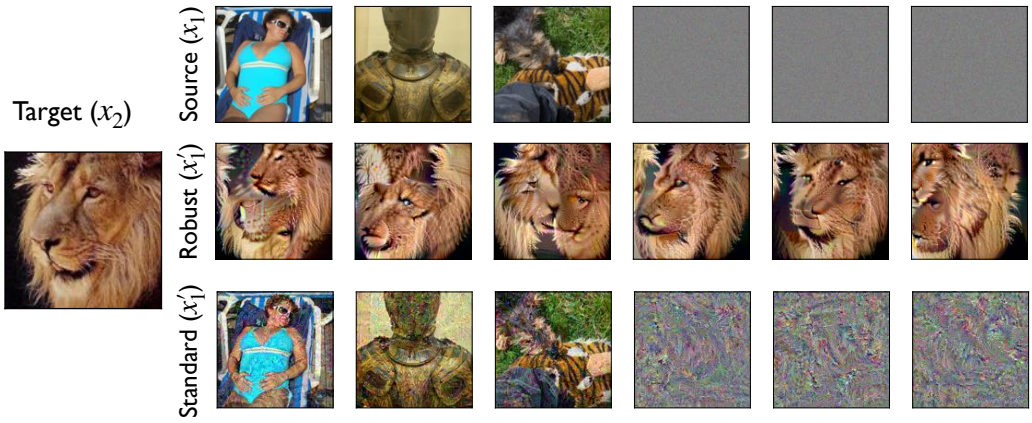


Figure A-24: Visualization of inputs that are mapped to similar representations by models trained on the ImageNet dataset. *Target (x_2) & Source (x_1)*: random examples image from the test set; *Robust* and *Standard* (x_1'): result of minimizing the objective (3.2) to match (in ℓ_2-distance) the representation of the target image starting from the corresponding source image for (*top*): a robust (adversarially trained) and (*bottom*): a standard model respectively. For the robust model, we observe that the resulting images are perceptually similar to the target image in terms of high-level features, while for the standard model they often look more similar to the source image which is the seed for the optimization process.

A.3.14 Image interpolations

Additional interpolations for robust models

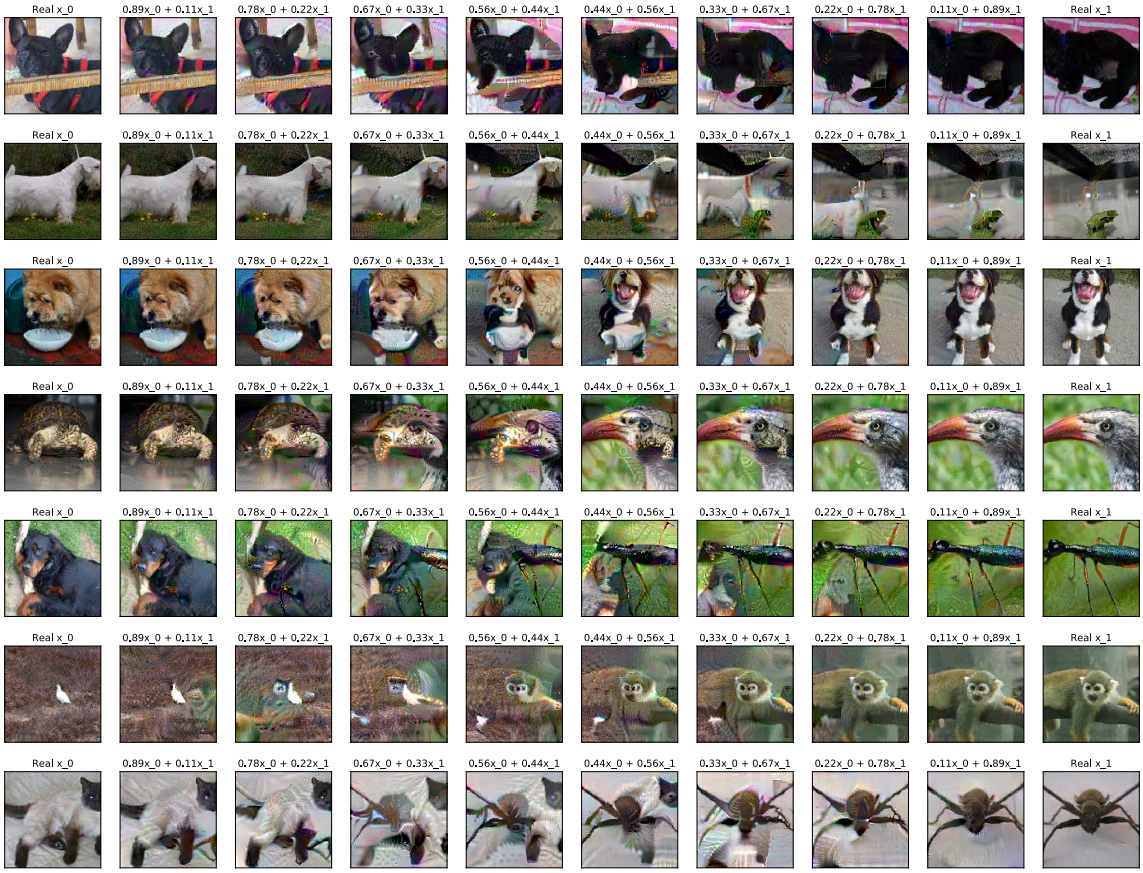


Figure A-25: Additional image interpolation using robust representations. To find the interpolation in input space, we construct images that map to linear interpolations of the endpoints in robust representation space. Concretely, for randomly selected pairs from the Restricted ImageNet test set, we use (3.1) to find images that match to the linear interpolates in representation space (A.19).

Interpolations for standard models

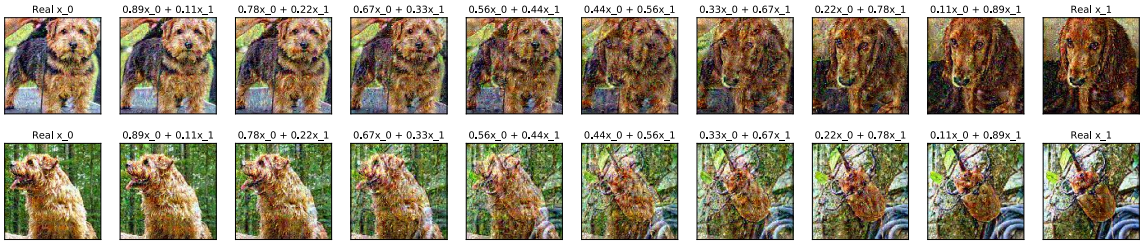


Figure A-26: Image interpolation using standard representations. To find the interpolation in input space, we construct images that map to linear interpolations of the endpoints in standard representation space. Concretely, for randomly selected pairs from the Restricted ImageNet test set, we use (3.1) to find images that match to the linear interpolates in representation space (A.19). Image space interpolations from the standard model appear to be significantly less meaningful than their robust counterparts. They are visibly similar to linear interpolation directly in the input space, which is in fact used to seed the optimization process.

A.3.15 Direct feature visualizations for standard and robust models

Additional feature visualizations for the Restricted ImageNet dataset

Figure A-27: Correspondence between image-level features and representations learned by a robust model on the Restricted ImageNet dataset. Starting from randomly chosen seed inputs (noise/images), we use a constrained optimization process to identify input features that maximally activate a given component of the representation vector (cf. Appendix A.2.7 for details). Specifically, (*left column*): inputs to the optimization process, and (*subsequent columns*): features that activate randomly chosen representation components, along with the predicted class of the feature.

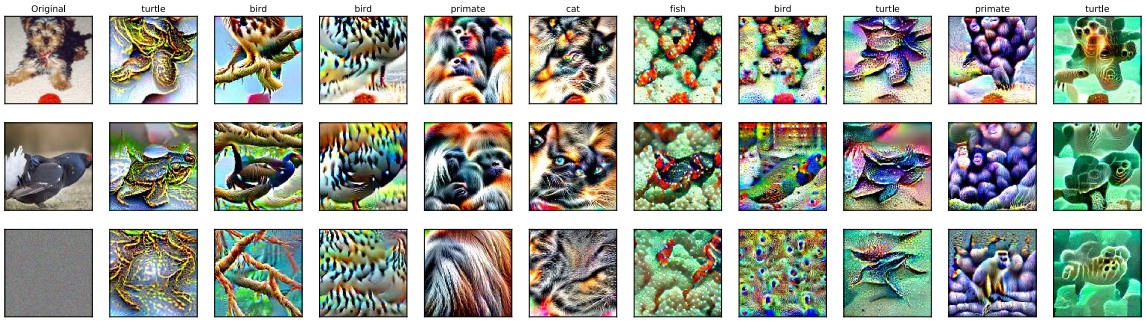


Figure A-28: Correspondence between image-level features and representations learned by a robust model on the Restricted ImageNet dataset. Starting from randomly chosen seed inputs (noise/images), we use a constrained optimization process to identify input features that maximally activate a given component of the representation vector (cf. Appendix A.2.7 for details). Specifically, (*left column*): inputs to the optimization process, and (*subsequent columns*): features that activate select representation components, along with the predicted class of the feature.

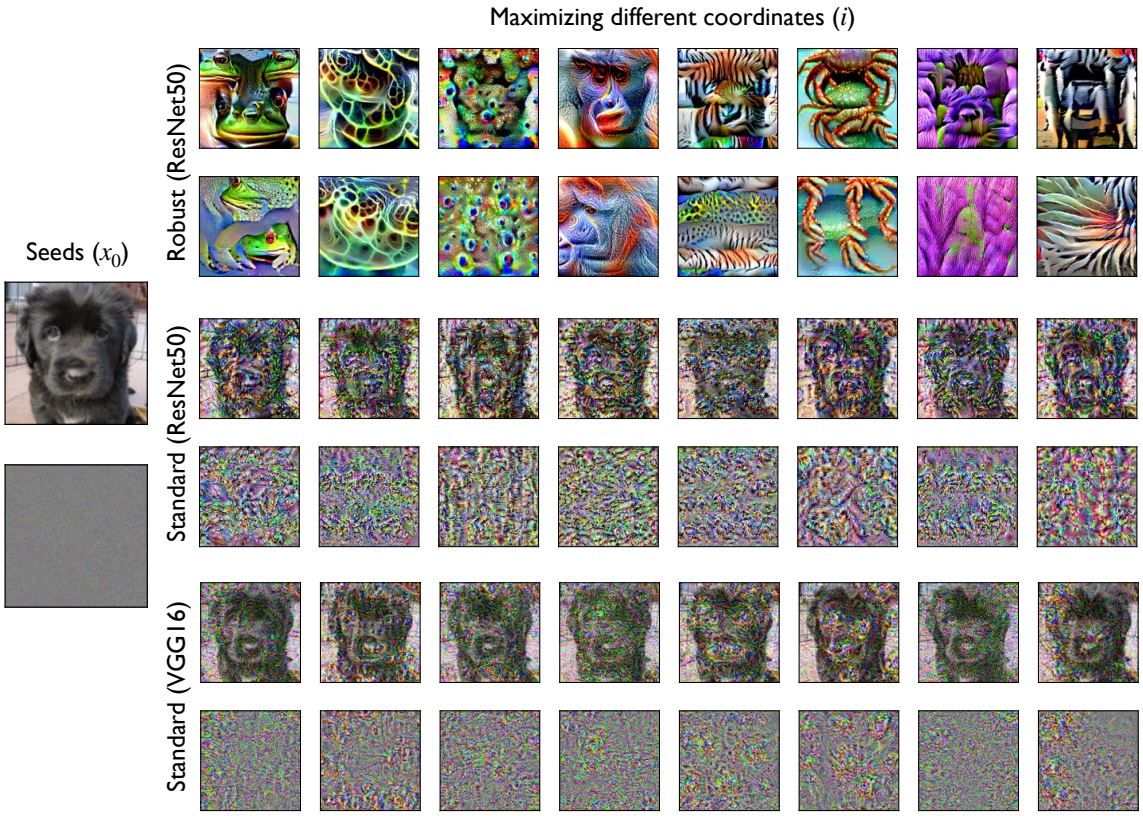


Figure A-29: Correspondence between image-level patterns and activations learned by standard and robust models on the Restricted ImageNet dataset. Starting from randomly chosen seed inputs (noise/images), we use PGD to find inputs that (locally) maximally activate a given component of the representation vector (cf. Appendix A.2.7 for details). In the left column we have the original inputs (selected *randomly*), and in subsequent columns we visualize the result of the optimization (3.3) for different activations, with each row starting from the same (far left) input for (*top*): a robust (adversarially trained) ResNet-50 model, (*middle*): a standard ResNet-50 model and (*bottom*): a standard VGG16 model.

Feature visualizations for the ImageNet dataset

Figure A-30: Correspondence between image-level patterns and activations learned by standard and robust models on the complete ImageNet dataset. Starting from randomly chosen seed inputs (noise/images), we use PGD to find inputs that (locally) maximally activate a given component of the representation vector (cf. Appendix A.2.7 for details). In the left column we have the original inputs (selected *randomly*), and in subsequent columns we visualize the result of the optimization (3.3) for different activations, with each row starting from the same (far left) input for (*top*): a robust (adversarially trained) ResNet-50 model, (*middle*): a standard ResNet-50 model and (*bottom*): a standard VGG16 model.

A.3.16 Additional examples of feature manipulation

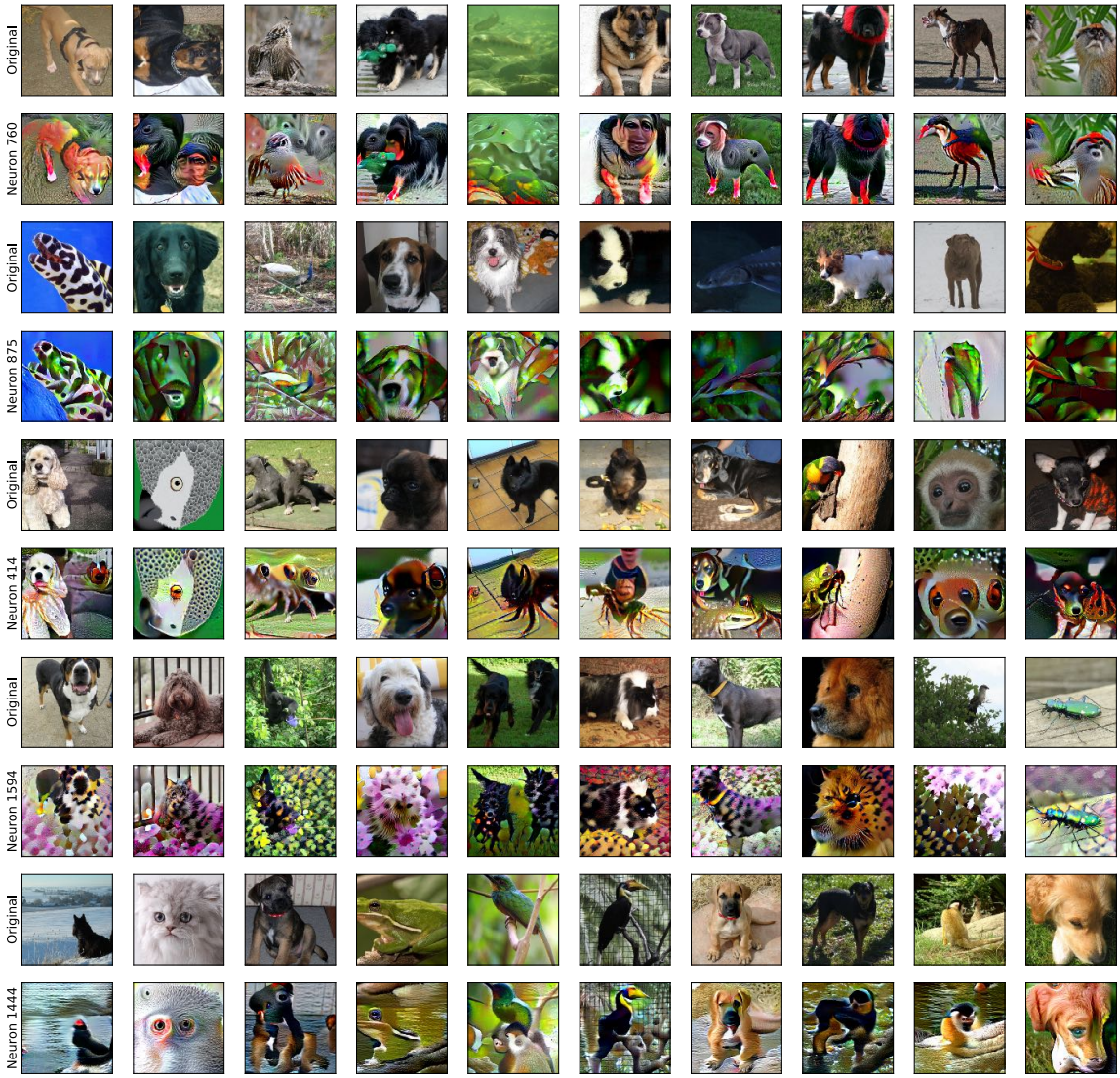


Figure A-31: Visualization of the results adding various neurons, labelled on the left, to randomly chosen test images. The rows alternate between the original test images, and those same images with an additional feature arising from maximizing the corresponding neuron.

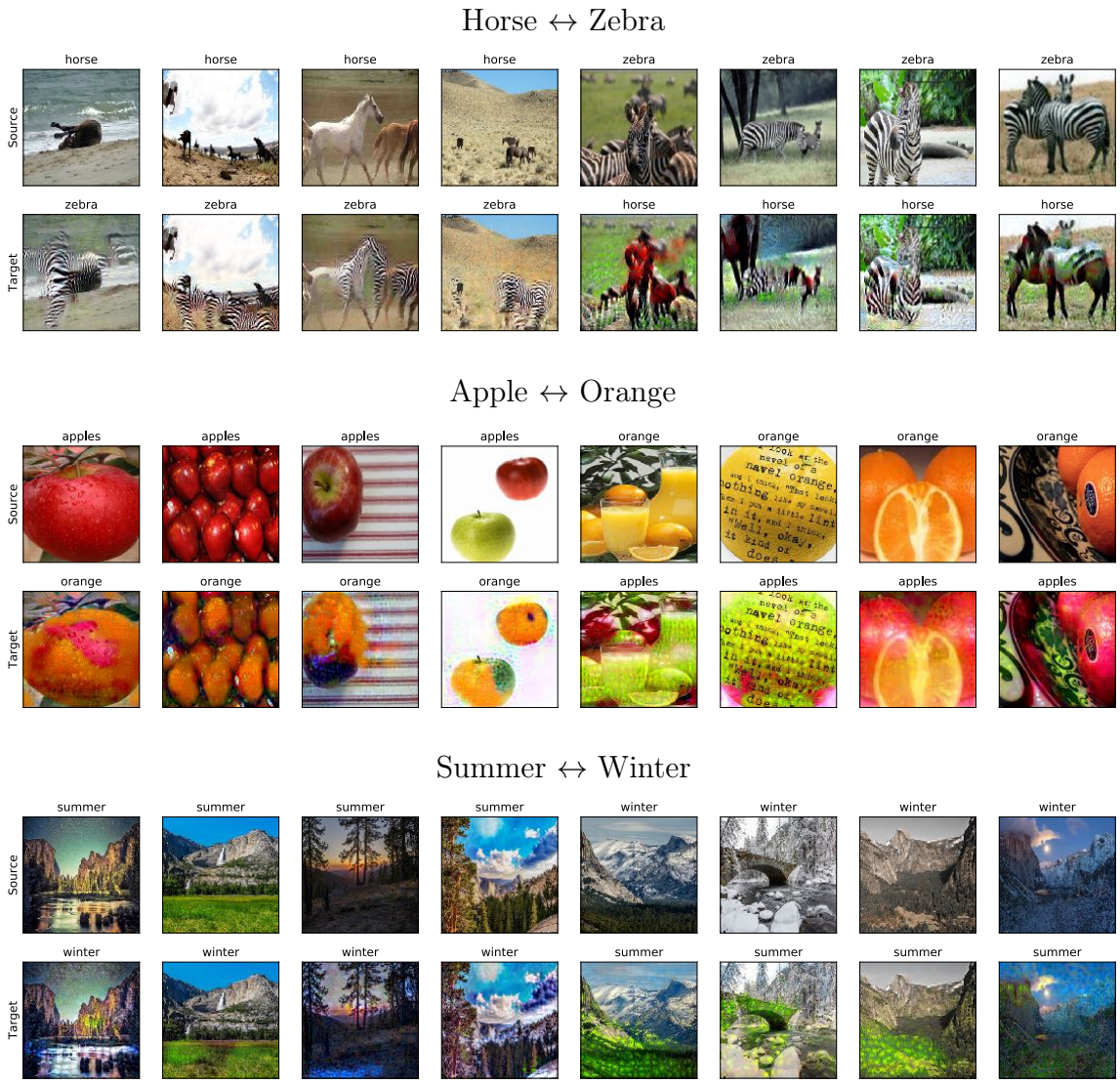


Figure A-32: Random samples for image-to-image translation on the Horse ↔ Zebra, Apple ↔ Orange, and Summer ↔ Winter datasets [Zhu+17]. Details in Appendix A.2.

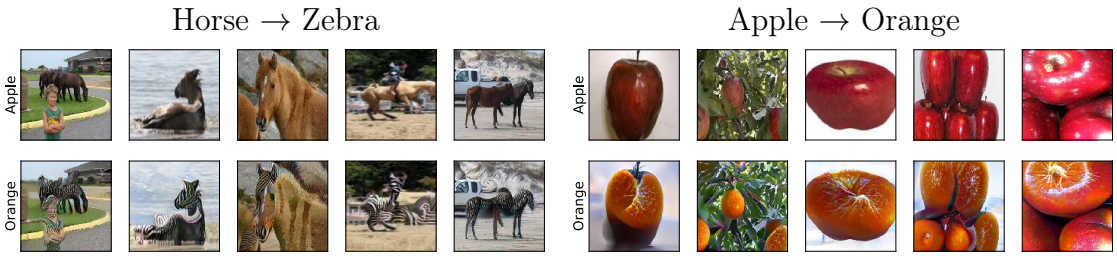


Figure A-33: Random samples for image-to-image translation on the Horse ↔ Zebra and Apple ↔ Orange datasets [Zhu+17] using the *same* robust model trained on the *entire ImageNet* dataset. Here we use ImageNet classes "zebra" (340) and "orange" (950).

Figure A-34: Training an ℓ_∞-robust model on the Horse ↔ Zebra dataset does not lead to plausible image-to-image translation. The model appears to associate "horse" with "blue sky" in which case the zebra to horse translation does not behave as expected.

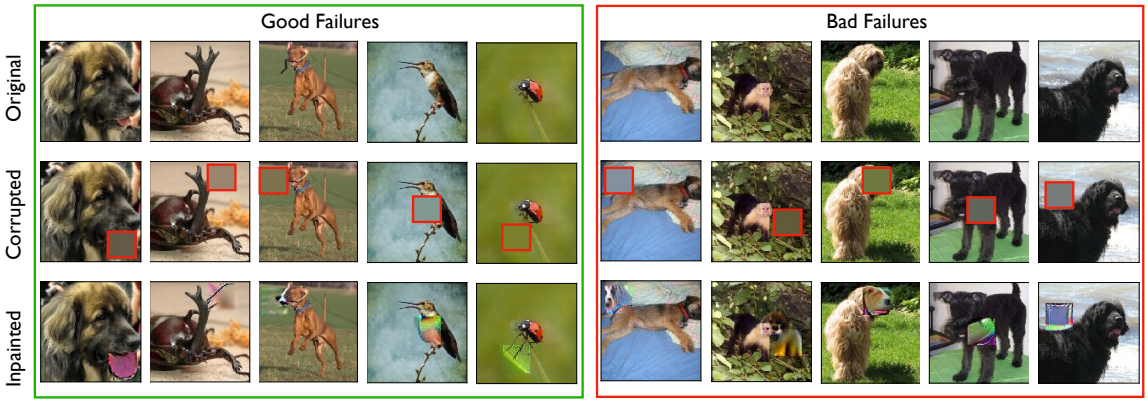


Figure A-35: Failure cases for image inpainting using robust models – *top:* original, *middle:* corrupted and *bottom:* inpainted samples. To recover missing regions, we use PGD to maximise the class score of the image under a robust model while penalizing changes to the uncorrupted regions. The failure modes can be categorized into "good" failures – where the infilled region is semantically consistent with the rest of the image but differs from the original; and "bad" failures – where the inpainting is clearly erroneous to a human.

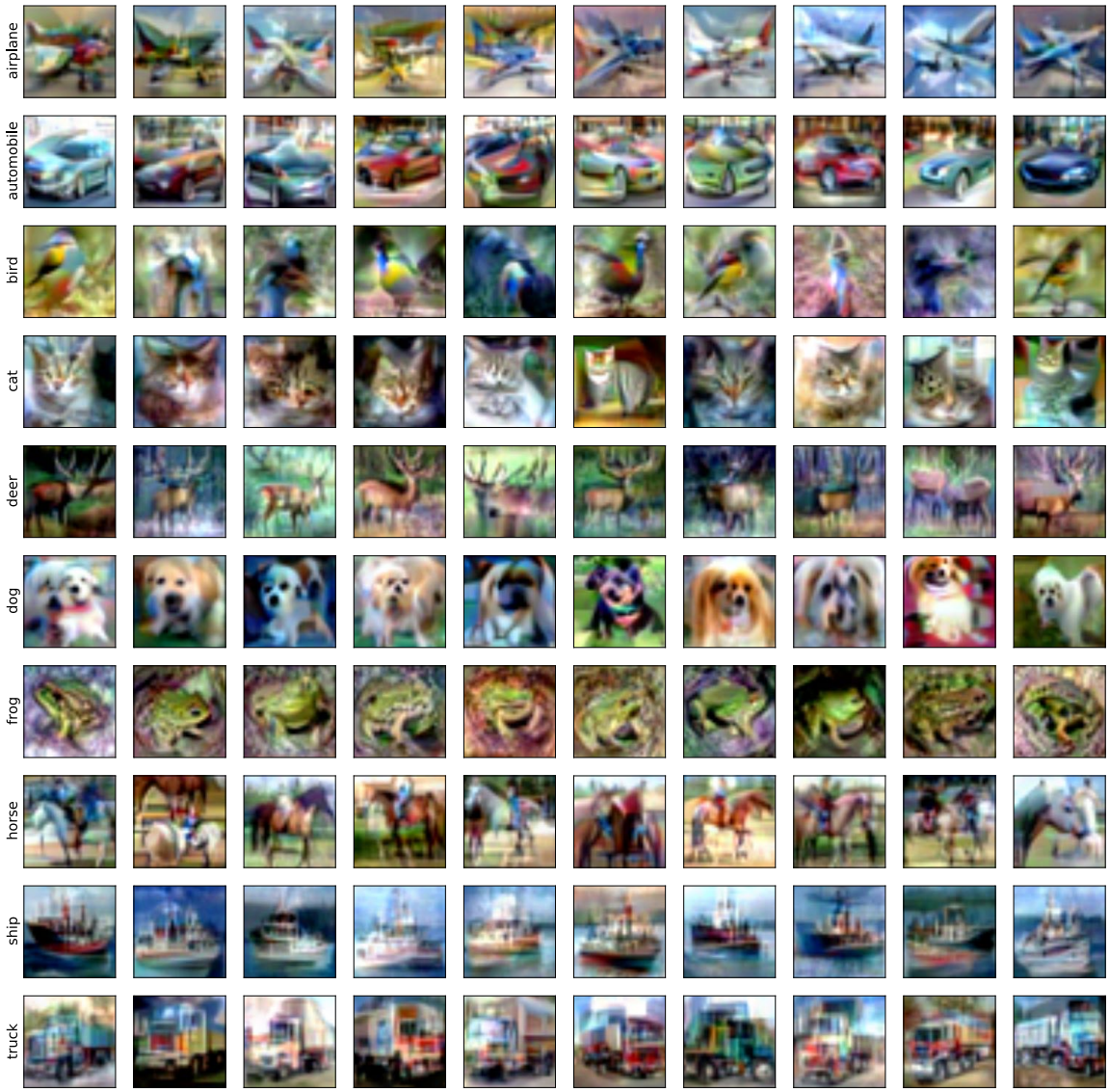


Figure A-36: Random samples generated for the CIFAR dataset.

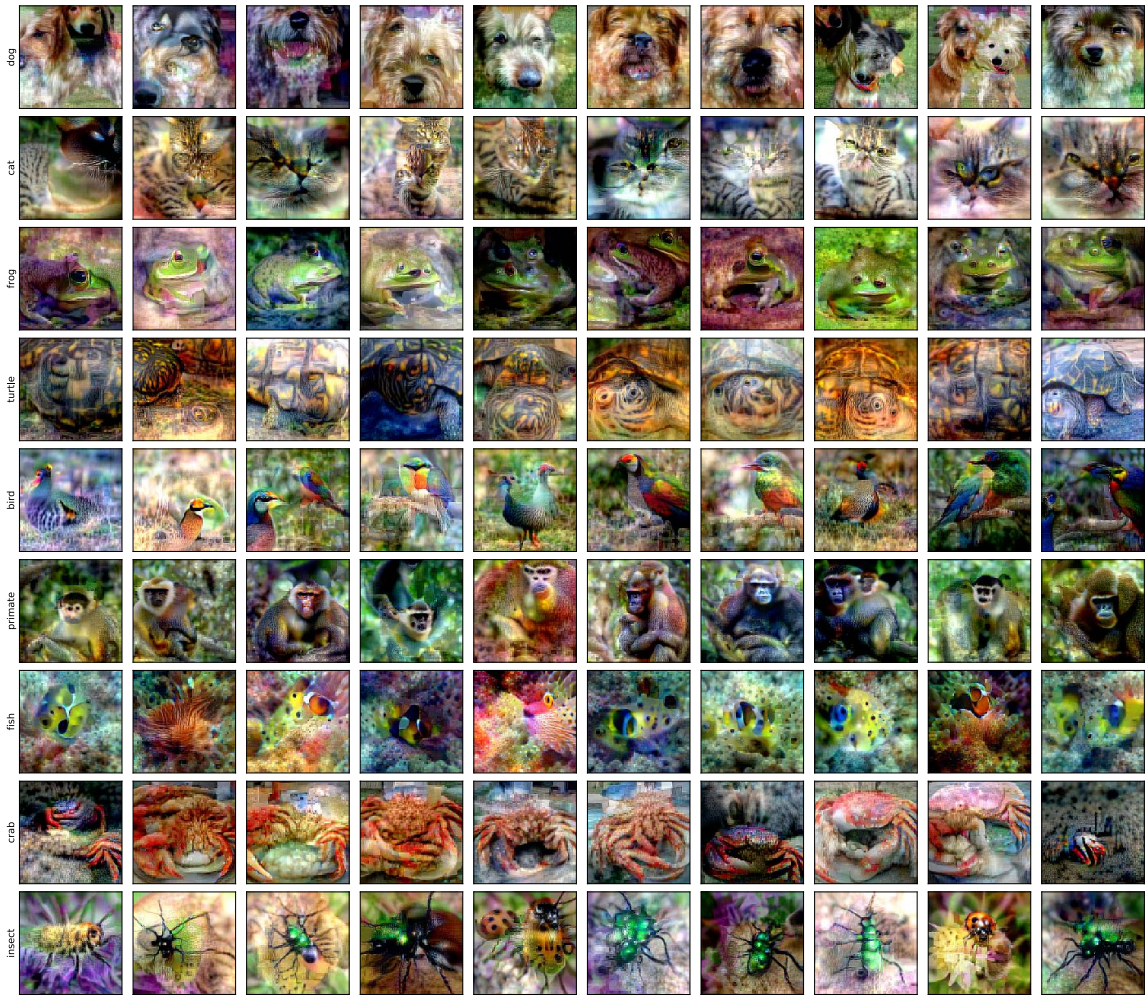


Figure A-37: Random samples generated for the Restricted ImageNet dataset.

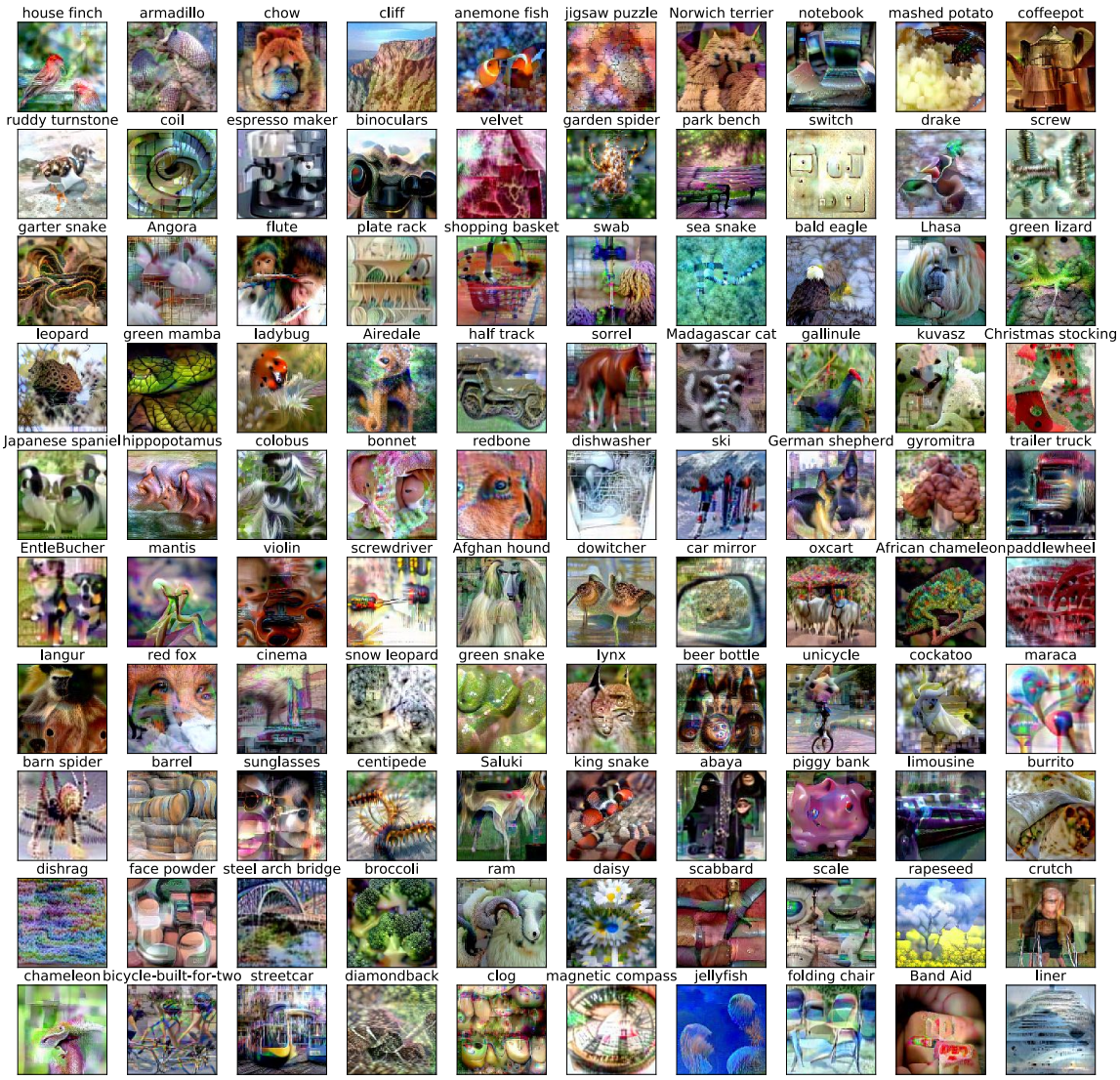


Figure A-38: Random samples generated for the ImageNet dataset.

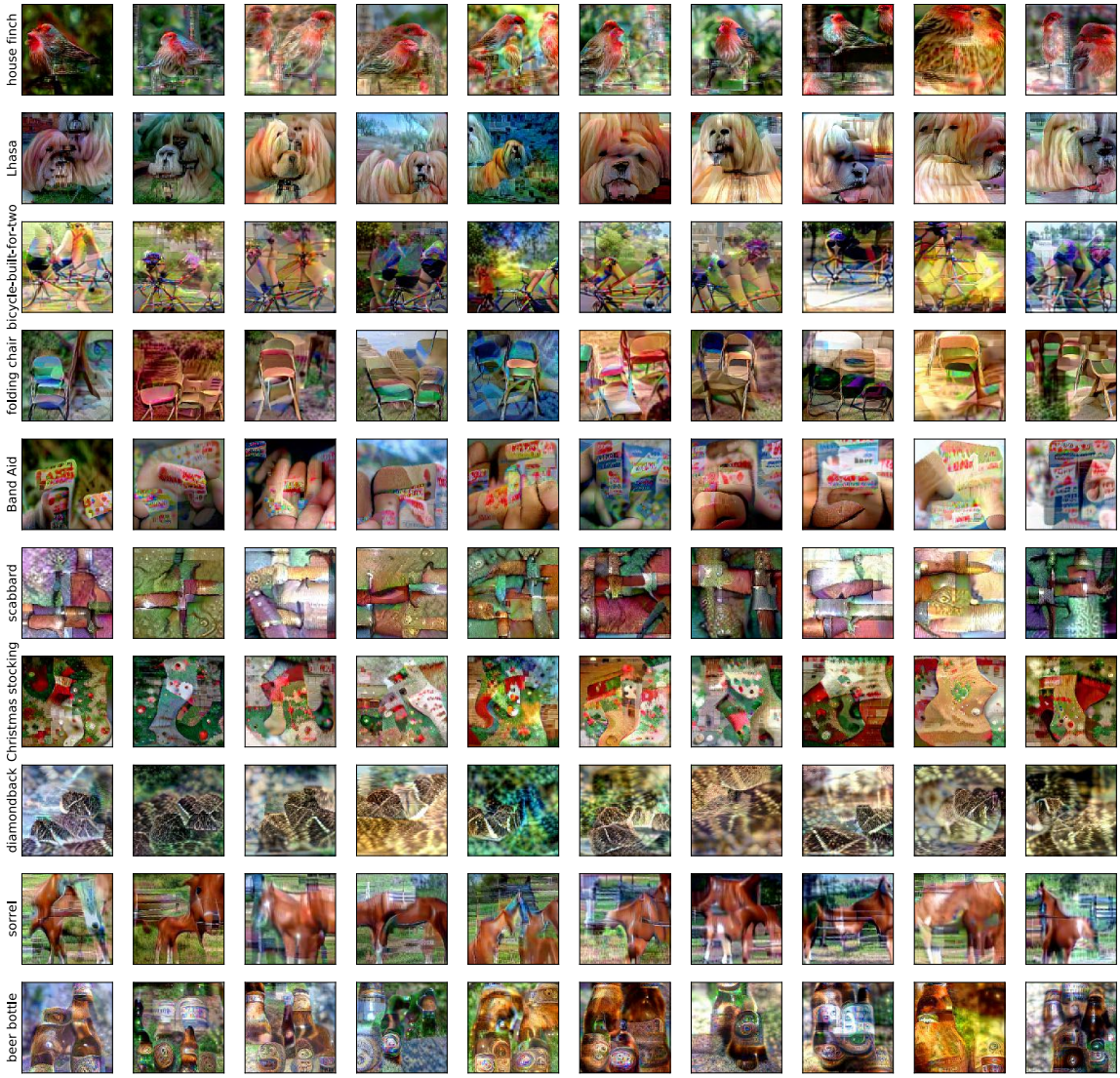


Figure A-39: Random samples from a random class subset.

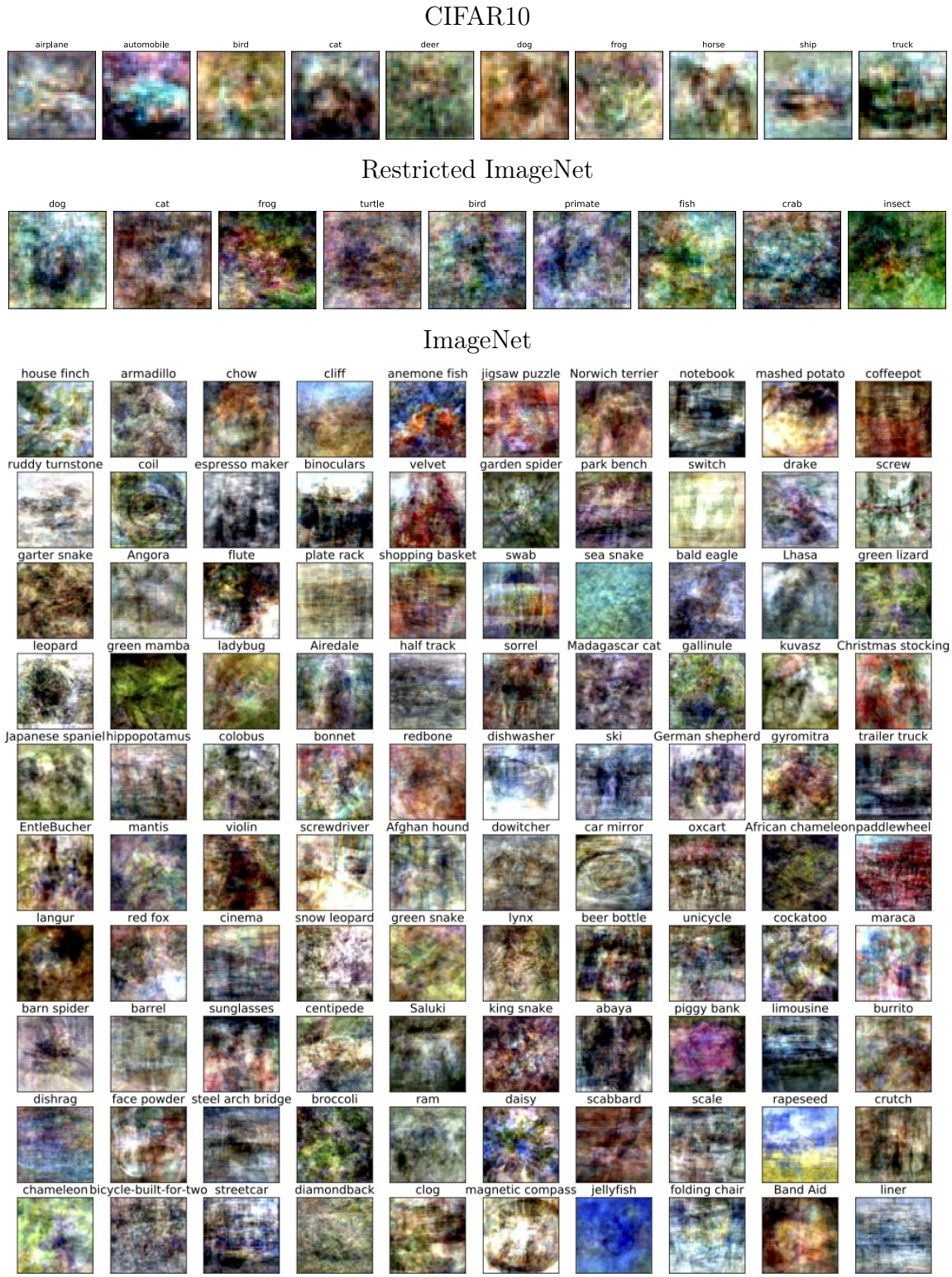


Figure A-40: Samples from class-conditional multivariate normal distributions used as a seed for the generation process.

A.4 Omitted proofs for backdoor data poisoning

A.4.1 Proof of Lemma 2

Proof of Lemma 2. We have

$$\langle X - \boldsymbol{\mu}_F, u\rangle = \langle X - \boldsymbol{\mu}_D, u\rangle + \varepsilon\langle \Delta, u\rangle \ , \text{ and}$$
$$\langle X - \boldsymbol{\mu}_F, u\rangle = \langle X - \boldsymbol{\mu}_W, u\rangle - (1-\varepsilon)\langle \Delta, u\rangle \ .$$

Thus if we let $t = \varepsilon\,|\langle \Delta, v\rangle| + \frac{\sigma}{\sqrt{\varepsilon}}$, we have

$$\Pr_{X\sim D}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| > t\right] \le \Pr_{X\sim D}\left[|\langle X - \boldsymbol{\mu}_D, v\rangle| > \frac{\sigma}{\sqrt{\varepsilon}}\right] \le \frac{\varepsilon}{\alpha^2} \ ,$$

by (4.1) and

$$\begin{aligned}\Pr_{X\sim W}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| < t\right] &\le \Pr_{X\sim W}\left[|\langle X - \boldsymbol{\mu}_W, v\rangle| > \langle \Delta, v\rangle - \frac{\alpha\sigma}{\sqrt{\varepsilon}}\right] \\ &\overset{(a)}{\le} \Pr_{X\sim W}\left[|\langle X - \boldsymbol{\mu}_W, v\rangle| > \frac{(\alpha-1)\sigma}{\sqrt{\varepsilon}}\right] \\ &\overset{(b)}{\le} \frac{\varepsilon}{(\alpha-1)^2} \ ,\end{aligned}$$

where (a) follows from assumption, and (b) follows from (4.2). □

A.4.2 Proof of Lemma 3

Proof of Lemma 3. By explicit computation, we have

$$\mathbb{E}_{X\sim D}\left[(X - \boldsymbol{\mu}_F)(X - \boldsymbol{\mu}_F)^\top\right] = \boldsymbol{\Sigma}_D + \varepsilon^2\Delta\Delta^\top \ ,$$
$$\mathbb{E}_{X\sim W}\left[(X - \boldsymbol{\mu}_W)(X - \boldsymbol{\mu}_W)^\top\right] = \boldsymbol{\Sigma}_W + (1-\varepsilon)^2\Delta\Delta^\top \ .$$

Therefore overall if $\boldsymbol{\Sigma}_F$ is the covariance of F, we have

$$\boldsymbol{\Sigma}_F = (1-\varepsilon)\boldsymbol{\Sigma}_D + \varepsilon\boldsymbol{\Sigma}_W + \varepsilon(1-\varepsilon)\Delta\Delta^\top \ ,$$

so in particular we have $\mathbf{\Sigma}_F \succeq \varepsilon(1-\varepsilon)\Delta\Delta^\top$. Thus $\|\mathbf{\Sigma}_F\|_2 \geq \varepsilon(1-\varepsilon)\|\Delta\|_2^2$.

Then we have

$$\begin{aligned}\varepsilon(1-\varepsilon)\|\Delta\|_2^2 &\leq v^\top \mathbf{\Sigma}_F v^\top \\ &= (1-\varepsilon)v^\top \mathbf{\Sigma}_D v + \varepsilon v^\top \mathbf{\Sigma}_W v + \varepsilon(1-\varepsilon)\langle v, \Delta\rangle^2 \\ &\leq \sigma^2 + \varepsilon(1-\varepsilon)\langle v, \Delta\rangle^2 \ .\end{aligned}$$

Since by assumption $\sigma^2 \leq \frac{\varepsilon}{6}\|\Delta\|_2^2$, we have

$$\langle v, \Delta\rangle^2 \geq \left(1 - \frac{1}{6(1-\varepsilon)}\right)\|\Delta\|_2^2 \overset{(a)}{\geq} \frac{2}{3}\|\Delta\|_2^2 \geq \frac{2\sigma^2}{\varepsilon} \ ,$$

where (a) follows by our assumption that $\varepsilon < 1/2$. The desired conclusion follows from taking square roots. □

A.4.3 A variation of Lemma 1 with finite sample bounds

As noted in Section 4.3, Lemma 1 as written may not directly apply to the situations we are interested in, simply because we ultimately care about applying this lemma to the training data. Thus to directly apply our guarantees to this setting, we require the same bounds, but over the empirical distribution. To do so, we use the following concentration bound:

Theorem 5 ([Ver18], Theorem 5.6.1)**.** *Fix $n \geq 1$. Let X be a random vector over $\mathbb{R}^d$, and assume $\|X\|_2 \leq K(\mathbb{E}[\|X\|_2^2])^{1/2}$ almost surely. Let $M = \mathbb{E}[XX^\top]$. Let $X_1, \ldots, X_n$ be m i.i.d. copies of X, and let $\widehat{M} = \frac{1}{n}\sum_{i=1}^n X_i X_i^\top$. Then, there exists a universal constant $C > 1$ so that with probability $99/100$, we have*

$$\|\widehat{M} - M\|_2 \leq C\left(\sqrt{\frac{K^2 d \log d}{n}} + \frac{K^2 d \log d}{n}\right)\|M\|_2 \ .$$

As a simple corollary of this, we can give a finite-sample version of Lemma 1:

Corollary 6. *Fix $1/4 > \varepsilon > 0$, and let $K > 0$. Let D, W be distributions over $\mathbb{R}^d$ with mean $\boldsymbol{\mu}_D, \boldsymbol{\mu}_W$ and covariances $\mathbf{\Sigma}_D, \mathbf{\Sigma}_W \preceq \sigma^2 I$, so that if $X \sim D$ (resp. $X \sim W$),*

then $\|X - \boldsymbol{\mu}_D\|_2 \leq K(\mathbb{E}[\|X - \boldsymbol{\mu}_D\|_2^2])^{1/2}$ *(resp* $\|X - \boldsymbol{\mu}_W\|_2 \leq K(\mathbb{E}[\|X - \boldsymbol{\mu}_W\|_2^2])^{1/2}$*) almost surely. Let* $F = (1-\varepsilon)D + \varepsilon W$ *be the mixture of* D, W *with mixing weights* $(1-\varepsilon)$ *and* ε*, respectively. Let* $X_1, \ldots, X_n$ *be* n *i.i.d. draws from* F*, where*

$$n = \Omega\left(\frac{d \log n}{\varepsilon}\right) .$$

Let $\mathcal{X}$ *be the subset of* $X_1, \ldots, X_n$ *drawn from* D*, and let* $\mathcal{W}$ *be the subset of* $X_1, \ldots, X_n$ *drawn from* W*. Then, if* $\|\boldsymbol{\mu}_D - \boldsymbol{\mu}_W\|_2^2 \geq \frac{10\sigma^2}{\varepsilon}$*, then if* $\boldsymbol{\mu}_F$ *is the mean of* $X_1, \ldots, X_n$ *and* v *is the top eigenvector of the empirical covariance, there exists* $t \in \mathbb{R}_{>0}$ *so that*

$$\Pr_{X \sim \mathcal{X}}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| > t\right] < \varepsilon$$
$$\Pr_{X \sim \mathcal{X}}\left[|\langle X - \boldsymbol{\mu}_F, v\rangle| < t\right] < \varepsilon ,$$

with probability at least $9/10$*.*

Proof. By a Chernoff bound, with probability $\geq 99/100$, we have that $|\mathcal{X}| \geq (1-2\varepsilon)n$ and $|\mathcal{W}| \geq \frac{\varepsilon}{2}n$. Condition on this event for the rest of the proof. Let $\widehat{\boldsymbol{\mu}}_D = \frac{1}{|\mathcal{X}|}\sum_{i \in \mathcal{X}} X_i$ be the empirical mean of the samples in $\mathcal{X}$, and let $\widehat{\boldsymbol{\mu}}_W$ be defined similarly for the samples in $\mathcal{W}$. Let $\widehat{\boldsymbol{\Sigma}}_D = \frac{1}{|\mathcal{X}|}\sum_{i \in \mathcal{X}}(X_i - \widehat{\boldsymbol{\mu}}_D)(X_i - \widehat{\boldsymbol{\mu}}_D)^\top$ be the empirical covariance of the points in $\mathcal{X}$, and let $\widehat{\boldsymbol{\Sigma}}_W$ be defined similarly for the samples in $\mathcal{W}$. Finally, let $\widehat{M}_D = \frac{1}{|\mathcal{X}|}\sum_{i \in \mathcal{X}}(X_i - \boldsymbol{\mu}_D)(X_i - \boldsymbol{\mu}_D)^\top$ be the empirical second moment with the actual mean of the distribution, and again define $\widehat{M}_W$ analogously. By Theorem 5, by our choice of n, we have

$$\|\widehat{M}_D - \boldsymbol{\Sigma}_D\|_2 \leq \frac{1}{4}\|\boldsymbol{\Sigma}_D\|_2 , \text{ and}$$
$$\|\widehat{M}_W - \boldsymbol{\Sigma}_W\|_2 \leq \frac{1}{4}\|\boldsymbol{\Sigma}_W\|_2 ,$$

both with probability at least $99/100$. Thus, by a union bound, all these events happen simultaneously with probability at least $9/10$. Condition on the event that all these

events occur. Then this implies that

$$\|\widehat{M}_D\|_2 \leq \frac{5}{4}\sigma^2 \ , \text{ and } \|\widehat{M}_W\|_2 \leq \frac{5}{4}\sigma^2 \ .$$

Since $\widehat{\mathbf{\Sigma}}_D \preceq \widehat{M}_D$ and $\widehat{\mathbf{\Sigma}}_W \preceq \widehat{M}_W$, we conclude that

$$\|\widehat{\mathbf{\Sigma}}_D\|_2 \leq \frac{5}{4}\sigma^2 \ , \text{ and } \|\widehat{\mathbf{\Sigma}}_W\|_2 \leq \frac{5}{4}\sigma^2 \ .$$

The result then follows by applying Lemma 1 to the empirical distributions over $\mathcal{X}$ and $\mathcal{W}$, respectively. $\square$