Identifying, Analyzing, and Addressing Weaknesses in Deep Networks

Foundations for Conceptually Sound Neural Networks

Klas Leino

CMU-CS-22-104

May 2022

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Matt Fredrikson, Chair
Anupam Datta
J. Zico Kolter
Corina Păsăreanu
Kamalika Chaudhuri

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under award number CNS-1801391, the National Security Agency under award number H9823018D0008, Intelligent Automation, Inc. under award number 24331, and the Air Force Office of Scientific Research under award numbers FA95501710600 and FA870215D0002. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Machine Learning, Artificial Intelligence, Security, Privacy, Robustness, Transparency, Neural Networks

For my family; my parents, India and Rustan, my siblings, Mattias, Yannika, and Kaleb, and my dog, Loki—who kept my lap warm the entire time I was writing this thesis.

Abstract

Deep neural networks have seen great success in many domains, with the ability to perform complex human tasks such as image recognition, text translation, and medical diagnosis; however, despite their remarkable abilities, neural networks have several peculiar shortcomings and vulnerabilities. Many of these weaknesses relate to a lack of *conceptual soundness* in the features encoded and used by the network—that is, the features the network learns to use may represent concepts that are not appropriate for the task at hand, even when they apparently allow the network to perform well on previously unseen validation data. This thesis examines the problems that arise in deep networks when they are not sufficiently conceptually sound, and provides steps towards improving the conceptual soundness of modern networks.

The first contribution of this thesis is a general, axiomatically justified framework for explaining neural network behavior, which serves as a powerful tool for assessing conceptual soundness. This work takes the unique perspective that to accurately assess the conceptual soundness of a model, an explanation must provide a *faithful* account of its behavior. By contrast, the literature has often attempted to justify explanations based on their appeal to human intuition; however, this begs the question, as it assumes the model captured conceptually sound human intuition in the first place.

To the contrary, a large body of prior work provides conclusive evidence that conceptual soundness is *not* the norm in standard deep networks, as *adversarial examples*—small, semantically meaningless input perturbations that cause erroneous behavior—found ubiquitously therein, violate the tenets of conceptual soundness. The second part of this thesis addresses this issue by contributing a state-of-the-art method for training neural networks with *provable guarantees* against a common class of adversarial examples.

Finally, we demonstrate that robustness to malicious input perturbations is only the first step—with contributions uncovering several orthogonal weaknesses and vulnerabilities relating to the conceptual soundness of deep networks.

Acknowledgments

The product of our experiences and interpersonal interactions is part of what makes us who we are today. In light of this, I would like to acknowledge the people who have made an impact on my trajectory by providing a brief account of how I arrived here.

Going to a typical public school, it would be easy to think math is boring, consisting mostly of tedious computations and memorization. Luckily, my dad—also a computer scientist—often brought home interesting math problems that he introduced to my siblings and me at the dinner table. Early on, I hardly recognized these problems as "math," but they were challenging and fun to puzzle over, giving me a greater appreciation for analytical thinking and problem-solving.

Like many talented children (I imagine), I was often getting into trouble at school, perhaps (I imagine) because the level of rigor and intellectual stimulation was insufficient to keep me engaged. One part that really captured my attention, however, was Mr. Burke's media tech class, where I learned to build websites, edit movies, and create animations in Flash. In particular, I enjoyed Flash enough that I asked for my own copy of the software for Christmas in 2006. With the ability to tinker around on my own time, I decided I wanted to make games in Flash. For this, I had to "learn to code," leading me to cobble together some ad hoc programming skills that were drawn primarily from Flash tutorials posted by an Italian blogger, Emanuele Feronato. This passion for programming Flash games sparked my interest in computer science—at a game making competition in high school, I took the advice of a Microsoft employee (whose name I unfortunately don't recall) to back up my game-making skills with a solid computer science foundation. Ultimately, this prompted me to apply to Carnegie Mellon.

As a high school senior, I had the privilege of interviewing with Frank Pfenning; and I am convinced that my opportunity to demonstrate the abilities I had learned through developing games ensured that I was not overlooked by an institution as selective as CMU's School of Computer Science.

In my undergrad years at CMU, I learned over time that I wanted to pursue the path of open-ended research, as opposed to going to industry as a software engineer. Finding the right research direction took some time; but among my classes, I particularly enjoyed theory-oriented topics like algorithm design, and the challenge of programming computer players from my game-making days had also led me to an interest in artificial intelligence. I began by studying computational geometry with Gary Miller—interestingly, I find my research occasionally drawing from ideas in high-dimensional geometry, a pleasant twist that I hadn't anticipated when I first transitioned to machine learning. I broadened my research experience with my senior thesis, advised by Guy Blelloch, where my project had elements of parallel algorithm design, low-level performance optimization, and machine learning. In my last semester, I took on another project

for Emma Brunskill and Manuela Veloso's graduate AI course, and Tai Sing Lee's computational neuroscience course, where I designed a novel method for image-tagging based on tracing activations in convolutional neural networks. This work felt particularly like a good fit for follow-up research, and was key to my ultimate decision to apply to grad school.

Back in Pittsburgh again after a year as a software engineer in New York—this time as a PhD student—I was contacted by Anupam Datta to sit in with the Accountable Systems Lab, due to our mutual interest in studying the inner-workings of deep networks. From here, I began working with Matt Fredrikson, and what followed is essentially documented in this thesis.

I would, of course, like to give special thanks to Matt for his mentorship; because of Matt's guidance, I explored many fascinating avenues that I might otherwise not have gone down—in particular, those related to security and privacy. During my time as a PhD student at CMU, I had the pleasure of working with many collaborators whom I wish to thank, including (alphabetically) Emily Black, Anupam Datta, Matt Fredrikson, Aymeric Fromherz, Jorge Guajardo Merchan, Jon Helland, Kaiji Lu, Ravi Mangal, Piotr Mardziel, Bryan Parno, Corina Păsăreanu, Shayak Sen, Nathan VanHoudnos, Saranya Vijayakumar, Zifan Wang, Sam Yeom, and Chi Zhang. Finally, I would like to thank my thesis committee—Matt Fredrikson, Anupam Datta, Zico Kolter, Corina Păsăreanu, and Kamalika Chaudhuri—for their time and input.

Overall, I'm grateful to all those I've learned from and worked with over the years, and for my friends and family who made this journey all the better to be on!

Contents

Chapter 1

Introduction

Deep neural networks have seen great success in many domains, with the ability to perform complex tasks such as image recognition, text translation, and medical diagnosis—tasks previously only humans have been considered adequately capable of. However, notwithstanding sensationalist claims of "human-level performance" on various tasks [56], modern neural networks remain prone to peculiar errors that humans would be unlikely to make. For example, neural networks have been found to be easily fooled by inputs that are carefully crafted to cause arbitrary classifications [111, 126, 141], to leak private information about the data they are trained on [43, 83, 128], and to place undue weight on weak or happenstantial correlations, leading to miscalibration and excessive bias [86]. These findings suggest that despite their degree of success, modern machine learning methods and models suffer from a multitude of weaknesses and vulnerabilities that require addressing if the limits of artificial intelligence are to be advanced.

In this thesis, we examine a class of weaknesses of deep learning connected by a common recurring theme: namely, a lack of what we will term *conceptual soundness* in the models produced by modern learning algorithms. That is, the features a network learns to use may represent concepts that are not appropriate for the task at hand, even when they allow the network to perform well on previously unseen validation data.

One of the powers of deep networks is their ability to extract structure from large volumes of raw, high-dimensional data in an unsupervised fashion. The layers of a deep network learn to encode progressively higher-level features that can be used for tasks, such as classification, without having to specify the concepts the network must use to perform its task effectively—or indeed how such concepts can even be represented from the primitive form of the data points it considers—*a priori*. While this has led to enormous strides in the capabilities of modern AI, it comes with a level of opacity and mystique that makes neural networks especially challenging to understand and analyze. In turn, the hidden risks of such systems are elusive. Without a better understanding of what a network has learned, we may rightfully be concerned about the quality of the learned features, and the way in which they are used.

Meanwhile, when networks indeed fail to be sound in the concepts they use, they conceal erroneous or otherwise compromising behavior that may go undetected when they are tested in tightly controlled circumstances resemblant of their training. The work presented in this thesis addresses these concerns on several fronts, introducing analysis techniques to determine when models are sufficiently trustworthy, studying the risks associated with when they are not, and providing steps towards achieving more conceptually sound models.

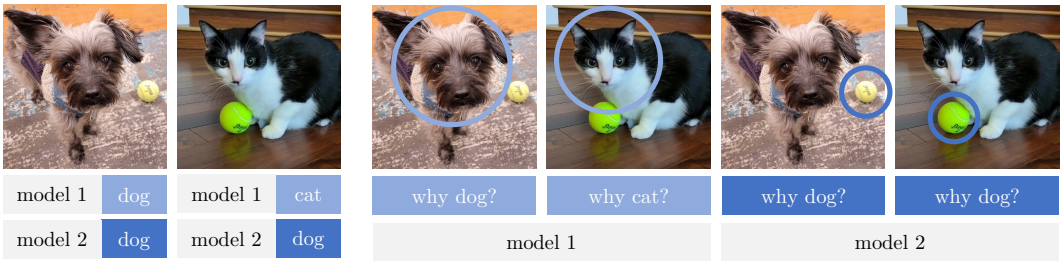


Figure 1.1: **(a)** predictions of two different models on an image of a dog and an image of a cat. We see that both models make the correct prediction on the dog image, but model 2 misclassifies model 2 as a dog rather than a cat. **(b)** possible explanations summarizing the internal logic used by each model to make its prediction on each image. The salient parts of the image (as determined by each model) are circled. We see that model 1 correctly uses the subject of the image to make its prediction, while model 2 erroneously focuses on the tennis ball in each picture, which it has associated with the dog class.

The presentation of this thesis is divided into three parts, summarized in greater detail in Sections 1.1–3. In Part I, we introduce the notion of conceptual soundness more concretely, and contribute a general and flexible framework for assessing the conceptual soundness of deep networks. Part II identifies the key role that a property called *robustness* plays in setting a foundation for conceptual soundness, and contributes a state-of-the art technique for producing robust neural networks. Finally, Part III contributes a novel analysis of two other key issues, separate from robustness, that relate to the conceptual soundness of a learned model. Specifically, we demonstrate how a lack of conceptual soundness compromises the calibration of the model and the privacy of the data used to train it.

1.1 Conceptual Soundness

Suppose we have two models that perform image classification. When presented with the picture of a dog shown in Figure 1.1a, both models correctly predict the class "dog." However, when we present each model with the picture of a cat also shown in Figure 1.1a, we find that the first model, *model 1*, correctly predicts "cat," while the second model, *model 2*, erroneously predicts "dog" once again.

Imagine that we could query models 1 and 2 in order to better understand how this bug comes about in model 2, and what model 1 does differently to avoid such a mistake. We may then find, as illustrated in Figure 1.1b, that model 1 arrives at its predictions naturally, by considering the dog that is the subject in the first image, and the cat in the second image. On the other hand, when we press the logic of model 2, the cause of the error is revealed: in both cases the model identifies a tennis ball in the picture, which it associates with the dog class.

Although tennis balls may indeed be strongly associated with dogs in the training data—and moreover, this may even reflect a true difference in dogs' and cats' respective natural affinities for tennis balls—we as humans understand that a tennis ball cannot be a determining factor for distinguishing between cats and dogs. Put differently, a tennis ball is not a *sound* concept for classifying cats and dogs. Thus, we will say that model 2 is not *concep-*

tually sound. By contrast, model 1 uses appropriate features to form its internal logic, and can therefore be considered conceptually sound.

Conceptual soundness is clearly a desirable property for any machine learning model. Even for models that apparently perform and generalize well, conceptual soundness is necessary for establishing model trust and integrity. For example, suppose that no images of cats with tennis balls (like the one in Figure 1.1) were seen by either model prior to deployment—not even as part of a held-out validation set. In such a scenario, models 1 and 2 may be indistinguishable using predictions on the available data alone. If model 2's focus on tennis balls is its only flaw, this may well be the case. On the other hand, when we peer into the inner-workings of the two models, the inadequacy of model 2 becomes readily apparent. Observing the conceptual soundness of model 1 boosts our confidence that it will handle new images correctly, and understanding the precise way in which model 2 is conceptually *unsound* on the picture of the dog enables us to anticipate the mistake it will make on the picture of the cat (or any hypothetical image like it) in advance.

In practice, it is entirely possible—and *likely*—that evidence of a model's lack of conceptual soundness will sometimes fail to arise in natural training and validation sets. Indeed, as we will see, the extensively studied ability for adversarial actors to fool neural networks into making arbitrary misclassifications is deeply related to a practically ubiquitous form of conceptual unsoundness that is admissible on "natural" in-distribution points.

In order to assess the conceptual soundness of a model, we need to form an understanding of *what concepts the model has learned*, and *how it uses them*. Like in the example in Figure 1.1b, we need an abstracted *explanation* of the model's "thought process," which we can then determine as either sound or unsound. This can be achieved using an *explanation framework*, which is tasked with succinctly but *accurately* summarizing a model's behavior on some part of its input space so that it can be better understood. Of course, deep networks are notoriously opaque and complex, meaning this task requires care. Moreover, while a conceptually sound deep network may yet be difficult to understand due to the nature of how it encodes its functionality, we cannot assume that a model is conceptually sound *a priori*, meaning that we must be able to distinguish explanations that are inadequate at meaningfully distilling a complex model's behavior from those that are symptomatic of the underlying model's unsoundness. That is, ultimately, *quality explanations require quality models*.

Part I of this thesis addresses the problem of exploring and evaluating conceptual soundness in deep neural networks. We begin by defining conceptual soundness in more concrete terms in Chapter 2, as it lies at the heart of our discussion throughout. Next, Chapter 3 contributes *Internal Influence*, a powerful and flexible explanation framework for deep networks that generalizes prior work while offering several unique capabilities that make it a particularly valuable tool for assessing conceptual soundness. Finally, Chapter 4 discusses the challenge of evaluating explanation frameworks themselves, offering a critique of some of the flavors of empirical tests that have been proposed for this in the literature, while arguing instead for a set of natural axioms to justify the explanations produced by a framework.

1.2 The Role of Robustness

Continuing our series of examples from Section 1.1, consider a third image classification model, *model 3*. Suppose that model 3, like model 1, does not make the mistake of encoding tennis balls as a feature for identifying dogs. However, it contains a perhaps more insidious

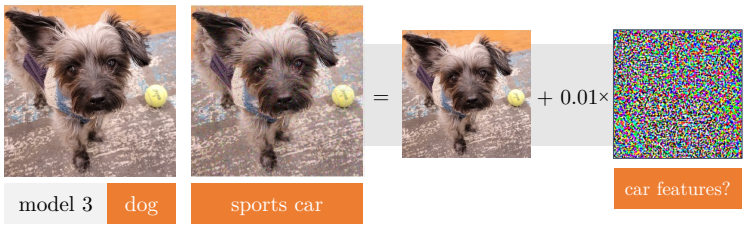


Figure 1.2: Prediction of a model on an image of a dog (left) and a perturbed image (right) derived by adding barely-perceptible noise-like features to the original image. We see that the model makes the correct prediction on the original image, but incorrectly predicts "sports car" on the perturbed image rather than "dog." This discrepancy suggests that that the model perceives the nose-like features as indicators of sports cars.

bug, illustrated by the example in Figure 1.2. In this example, model 3 is presented with two images. The first, on the left is the same dog image from our previous example, which model 3 correctly labels. The second image appears visually the same, but has been perturbed slightly with a pattern that, even when amplified to be detectable by the human eye, looks essentially like noise. On this perturbed image, model 3 perplexingly predicts "sports car," a clearly incorrect label entirely unrelated to the contents of the image.

Taking the performance implications of this strange bug aside for a moment, consider what this example entails for the conceptual soundness of model 3. First, how can the model's prediction of "sports car" on the second image be explained? Given that the added perturbation led the model to change its prediction from "dog" to "sports car," it seems the model treats these noise-like features as important indicators of sports cars, constituting a clear violation of conceptual soundness.

Conversely, consider the explanation for model 3's correct prediction of "dog" on the first image. At some level of abstraction, it is possible that model 3 identified sound dog features; however, more locally, it applies less intuitive logic. Namely, *precisely because each of the pixels did* ***not*** *take their nearby perturbed values from the second image*, model 3 predicted "dog," when it could easily have predicted "sports car." This may seem like a facetious explanation—there are many features besides the noise-like perturbation that are *not* present in the first image—however, the perturbation (or lack thereof) is particularly salient because it represents such a small change to the input, meaning model 3 is highly sensitive to its presence. In other words, the perturbation represents a highly important direction (for both dogs and sports cars) in which the model's output rapidly changes.

The perturbed image from Figure 1.2 constitutes an instance of what has become known in the literature as an *adversarial example*, recognized to be first reported on by Szegedy et al. [141]. As in our illustration in Figure 1.2, an adversarial example is an input to a model that resembles one class (e.g., "dog"), while being classified as another (e.g., "sports car") by the model. While the concept of "resemblance" is nebulous, we typically take it to mean that an adversarial example is derived by perturbing a natural input in a *semantically meaningless* way—for example, the perturbation may be small enough to be imperceptible to the human eye (like in Figure 1.2), or simply inconspicuous in the given context.

Adversarial examples impact the reliability of neural networks that are vulnerable to them—and constitute a security concern in safety-critical machine learning systems—as they lead to unexpected erroneous behavior on seemingly benign inputs. Moreover, as discussed, they represent a fundamental hurdle for conceptual soundness. As such—and due to their

ubiquity in standard neural networks [19, 48, 111, 141]—the problem of adversarial examples is of central importance to modern machine learning.

To defend against adversarial examples, we often aim to obtain so-called *robust* models, that are resistant to malicious perturbations. Methods for improving model robustness usually consider a specific class of adversarial examples that can be precisely defined without depending on an ill-defined notion of human perception, namely *small-norm* adversarial examples. As the name suggests, a small-norm adversarial example is one for which the distance (according to some metric, e.g., Euclidean distance) between the original point and its perturbed counterpart is below some small threshold, typically denoted by ϵ. In terms of perception, when ϵ is sufficiently small, any points that are ϵ-close to the original input will be perceptually indistinguishable from the original input, meaning that small-norm adversarial examples can be properly considered a subset of the broader notion of perturbations that preserve the "resemblance" of a point.

We say that a model is *locally robust* at a point, x, if all points within a distance of ϵ from x receive the same label as x from the model. Hence, we see that small-norm adversarial examples cannot be derived from points for which a model is locally robust. We may thus informally refer to a model that resists adversarial examples by achieving local robustness at as many points as possible as a *robust model*.

Part II of this thesis is dedicated to addressing the challenge of robustness as an important cornerstone of conceptual soundness. Chapter 5 introduces adversarial examples, robustness, and their relation to conceptual soundness more formally. Chiefly, we conclude in this chapter that *robustness is necessary for conceptual soundness*. Next, in Chapter 6, we delve into our contributions to the literature on training models that achieve *provable guarantees* of robustness. Specifically, we propose an elegant and effective defense that modifies the architecture of a neural network to naturally provide certificates of local robustness without introducing overhead to the network's inferences. Finally, Chapter 7 discusses limitations of local robustness, demonstrating that in some contexts it is too stringent (we propose some natural relaxations), while at the same time it is insufficient to ensure conceptual soundness.

1.3 Weaknesses Related to Conceptual Soundness

As we have seen, conceptual soundness is a desirable property for reliable neural networks. Furthermore, while robustness serves as an important step in achieving conceptual soundness, it is not sufficient to guarantee that the model will use features appropriately. Indeed, one need look no further than our original example in Figure 1.1 to find an illustration of this point. Namely, it is entirely possible that model 2, which erroneously conflates dogs with tennis balls, is a perfectly robust model—at least in the sense of *local robustness* described in Section 1.2 prior. That is, perhaps model 2 cannot be fooled via small perturbations into changing its perception of the high-level features it observes. Even carefully-crafted manipulations may not lead the model astray from correctly identifying tennis balls, dog ears, cat noses, etc. Nonetheless, if it sees a tennis ball it will insist that it has been presented with a picture of a dog regardless of the other features available.

In practice, numerous issues may arise in deep networks that are intrinsically tied to conceptual soundness, but not necessarily to robustness. In the final part of this thesis, we embark on a deep dive into two such issues. The first, which we will focus on most heavily, is the concern of data privacy. Essentially, a model that is too closely tailored to

its training set may unwittingly leak private information contained in the training data. A particular consequence of this issue, which is studied much more broadly in the field of machine learning, is *overfitting*, wherein a model behaves differently on points from its training set than on novel instances. Second, we will examine how a model may learn to place undue weight on weak or happenstantial correlations, leading to miscalibration and excessive bias, a phenomenon that has been termed *feature-wise bias amplification* [86].

Overfitting and Privacy Risks

Overfitting is a central problem in machine learning that is strongly tied to the reliability of a learned model when it is deployed on unseen data. Overfitting is often measured—or even defined—by the difference in accuracy obtained by a model on its training data, compared to on previously unseen validation data (i.e., the generalization error). While this is a useful metric that broadly captures the extent to which a model will make mistakes on new points (one of the key problematic implications of overfitting), we will instead take a more general and nuanced take on overfitting. In particular, we will examine a key mechanism underlying overfitting: the encoding and use of unsound features; i.e., the lack of conceptual soundness.

In essence, deep networks function by learning to extract high-level features that enable them to make predictions on new inputs. While some of these features may truly be generalizable, conceptually sound predictors, others may be learned simply because they coincidentally aid classification on the training set only. The latter type of learned features are not conceptually sound, and thus may lead to anomalous or incorrect behavior on unseen points, i.e., overfitting. For example, in our example from Section 1.1, model 2 learned to associate tennis balls with dogs (presumably because its training set featured this association), leading to the mistake we observed when the model was presented with a cat with a tennis ball.

It is worth taking note that this may not have been measured as overfitting in the classical sense (i.e., measurable generalization error) if no instances like the cat-tennis-ball example were included in the validation set used to calculate the generalization error. Moreover, while overfitting through the use of a conceptually unsound, but possibly generally occurring feature (like a tennis ball) may indeed fail to be measured as generalization error, more subtle, yet possibly more egregious cases of overfitting may be even more likely to go undetected on sample data. For example, if some image in the training set had been directly memorized as a feature itself, the effect of such blatant overfitting may be kept highly localized to the memorized point. Thus, in addition to potentially causing misclassifications, overfitting presents a privacy risk. Namely, by learning features that are overly-specific to the training set, models will inadvertently leak information about their training data. The work presented in Chapter 8 uses insights in line with those presented here to show how a range of different types of attackers can make various sorts of inferences about the data used to train a model.

Feature-Wise Bias Amplification

Bias amplification occurs when the distribution over a model's prediction outputs is more skewed in comparison to the prior distribution of its prediction target. For example, suppose we have a fourth model, *model 4*, performing the same image classification task used in our illustrations throughout this chapter. Suppose further that 75% of the images presented to model 4 would always be cats (perhaps due to the abundance of cat videos on the internet). In this case, model 4 would exhibit bias amplification if, for example, 90% of

its predictions come back as "cat" (amplifying the 75% prior). This phenomenon has been observed and reported on in the context of several real classification tasks in the recent literature [86, 139, 171].

Intuitively, bias amplification is always undesirable, since it necessarily implies imperfect accuracy. In the strictest sense, however, this is not entirely true: as we will see, bias amplification may be *unavoidable* in contexts where the available features are not sufficiently informative. Nevertheless, in practice, bias amplification has been reported in contexts where the features presumably contain sufficient information to classify accurately, suggesting that it is in fact detrimental to the model's utility. In Chapter 9 we show that indeed, a particular form of strictly detrimental bias amplification may arise that can be attributed to a suboptimal weighting of features by the learning process—we call this *feature-wise bias amplification*. This form of bias amplification constitutes a violation of conceptual soundness, as it implies that the model does not use features appropriately.

Part I

Evaluating Conceptual Soundness

Chapter 2

Explanations and Conceptual Soundness

Conceptual soundness, a theme we will recurrently visit throughout this thesis, is paramount to establishing model trust and avoiding a myriad of potential weaknesses and vulnerabilities in deep networks. While we previously introduced conceptual soundness informally—as the proper use of appropriate features by a model—we now turn to make this notion more concrete in this brief chapter. Section 2.2 presents a formalization of conceptual soundness, which we state in terms of an *explanation framework*, formally defined in Section 2.1.

2.1 Explanation Frameworks

Typically, it is most useful to think of a neural network abstractly as a function, $f : \mathbb{R}^n \to \mathbb{R}^m$, where n and m represent the input and output dimensions, respectively. Primarily, the work in this thesis will consider neural network *classifiers*, which categorize their inputs into a discrete set of classes. In such cases, m represents the number of classes, and each dimension of f's output corresponds to a *logit value*, or unscaled probability score, for a particular class. These logit values are often scaled to a probability space and interpreted as confidences. We will find it useful to distinguish between the network's (discontinuous) prediction function, $F : \mathbb{R}^n \to [m]$, and (continuous) logit function f, using an upper/lower-case notation. The predictions can be derived by taking the index of the maximal logit output from f, i.e., $F(x) = \text{argmax}_i\{f_i(x)\}$.

The types of functions neural networks are employed to capture often correspond to high-level human tasks, making it unclear how one would explicitly construct them. The "magic" of neural networks comes from their ability to learn such complex functions from simple parameterizations and learning objectives (and lots of data). The resulting systems may achieve impressive results, but are intrinsically opaque as their parameterizations do not aid particularly in forming an understanding of the nature of the function they represent. Thus, in order to gain transparency and insight into our learned models, it is helpful to have the ability to probe their behavior through an *explanation framework*. As suggested by the discussion in Chapter 1, explanations provided by such frameworks allow us to diagnose potential problems in models that might otherwise elude us, and to increase our confidence in models that prove to act on sound principles.

Formally, an explanation framework, which we will denote by $\chi : (\mathbb{R}^n \to \mathbb{R}^m) \to \Psi \to \Omega$, is a function from a model, $f : \mathbb{R}^n \to \mathbb{R}^m$, and a space of explanation parameters, Ψ, to a space of possible explanations, Ω. As an illustrative example to understand what these components mean, we will recall the explanations from Figure 1.1b in Chapter 1. In this example, explanations were produced as a function of a given model and image. Concretely then, the corresponding parameter space would be $\Psi = \mathbb{R}^n$, the space of inputs to the model. The resultant explanations consisted of a circled region superimposed onto the provided image, indicating the most salient region of the original image contributing to the model's prediction. Thus, the corresponding explanation space for this framework could be defined as the set of all circles that could be drawn onto the provided image. More explicitly, this could be represented as $\Omega = \mathbb{R}^3$, the set of 3-tuples consisting of an x-coordinate, y-coordinate, and radius of the superimposed circle.

For the remainder of this chapter, we will consider this toy explanation framework as a running example. The precise function χ in $(\mathbb{R}^n \to \mathbb{R}^m) \to \mathbb{R}^n \to \mathbb{R}^3$ that produces the explanations is not important, and can be considered a black box for now. We will revisit explanation frameworks again in more detail in Chapter 3 when we introduce *Internal Influence* for influence-directed explanations [85], a real-world explanation framework.

As a final remark, we have not, as of yet, described the requisite properties of χ that would allow us to consider its outputs to be valid "explanations" of the provided model. Clearly, it would seem improper to refer to just any arbitrary function in $(\mathbb{R}^n \to \mathbb{R}^m) \to \Psi \to \Omega$, as an explanation framework—at the very least the purported explanations should be tied in some meaningful way to the model f and the explanation parameters ϕ. We will refer to this property as *faithfulness*; intuitively, an explanation framework is *faithful* to the model f if it produces explanations that are causally related to f. We will define the concept of faithfulness more rigorously in Chapter 4 when we provide an axiomatic justification for Internal Influence.

2.2 Conceptual Soundness

We now turn to defining the notion of conceptual soundness more explicitly. We will do so in terms of a set of "desired" explanations that we would be satisfied with obtaining on an ideal model. More formally, let $\Omega^* : \Psi \to 2^\Omega$ be an *explanation criterion*. Essentially, Ω^* maps possible parameterizations of χ to a set, $\Omega^*(\phi) \subseteq \Omega$, of admissible explanations for that parameterization. In terms of our example explanation framework from Section 2.1, Ω^* specifies a set of acceptable superimposed circles for each possible image. For example, on the dog image from Figure 1.1, we may allow a range of circles that capture the the dog from the image; however, we would *not* admit circles that focus on the tennis ball, or any background part of the image.

There may be certain parameterizations that do not make sense to consider—e.g., an image consisting entirely of noise. For such parameterizations, we can simply let $\Omega^*(\phi) = \Omega$, indicating that no requirements are imposed on the explanations produced on these parts of the parameter space.

Using the concept of an explanation criterion, we are now ready to provide a formalization of conceptual soundness, stated in Definition 2.1. Intuitively, this definition states that a model is conceptually sound if the explanations produced for the model are consistent with our desired explanations.

Definition 2.1 (Conceptual Soundness)**.** *Given a faithful explanation framework,* $\chi : (\mathbb{R}^n \to \mathbb{R}^m) \to \Psi \to \Omega$*, and an explanation criterion,* $\Omega^* : \Psi \to 2^\Omega$*, we say that a model,* $f : \mathbb{R}^n \to \mathbb{R}^m$*, is* conceptually sound *if*

$$\forall \phi \in \Psi \quad . \quad \chi(f \ ; \ \phi) \in \Omega^*(\phi)$$

While Definition 2.1 is stated universally over the entire parameter space, it should be clear that it also makes sense to refer to a model as conceptually sound in some parts of the space—i.e., those for which $\chi(f \ ; \ \phi) \in \Omega^*(\phi)$—and not others. In other words, Definition 2.1 can be extended to capture partial conceptual soundness.

To match the informal intuition we have presented for conceptual soundness previously, we note that conceptual soundness should principally be a property of a *model*. That is, whether or not a model is conceptually sound should be a reflection of the model itself. The explanation framework through which the model is defined should not artificially lead to conceptual soundness, nor should a lack of conceptual soundness be an indictment of the underlying explanation framework. The former is ensured if the explanation framework is faithful to the model, which essentially entails that the model must earn its desired explanations rightfully.

As to the latter point, we see that a model may fail to be conceptually sound when the explanation criterion is not achievable within the given explanation framework. To avoid this setting, conceptual soundness can only be considered meaningful in cases where the explanation framework is sufficiently expressive (Definition 2.2) for our given requirements.

Definition 2.2 (Sufficient Expressiveness)**.** *We say that an explanation framework,* $\chi : (\mathbb{R}^n \to \mathbb{R}^m) \to \Psi \to \Omega$ *is* sufficiently expressive *for explanation criterion,* $\Omega^* : \Psi \to 2^\Omega$ *if* $\exists f \in (\mathbb{R}^n \to \mathbb{R}^m)$ *such that* f *is conceptually sound with respect to* χ *and* Ω^**.*

We will typically assume that Definition 2.2 holds, meaning that our desiderata are "reasonable" for the context they are given in.

Chapter 3

Influence-Directed Explanations

When a deep neural network makes a decision, we would like to know that we can trust it. Did our model make a connection between spuriously-correlated events? Did our model learn a pattern that we overlooked but might find useful? If our model made a mistake, why? Answering these high-level questions requires the ability to make a rich set of queries that help us learn about a model's predictive behavior and assess its conceptual soundness. Primarily, an explanation framework is meant to give us the means to make such queries; its utility comes from its ability to express and accurately answer queries.

As compared to inherently interpretable machine learning models, such as simple decision trees or linear classifiers, deep networks are capable of far more complex behavior, but operate opaquely. As the power of deep networks is often required for the most challenging tasks, if we additionally desire transparency, we are left with the task of understanding deep network behavior post hoc. This should be seen as a scientific process; the network's full behavior cannot necessarily be summarized succinctly, but we can aid our understanding of its most important characteristics through carefully-tailored queries that enable hypothesis formulation, evaluation, and refinement. One explanation may not be sufficient to totally "understand" a neural network, but it should provide insight that sharpens our conception of it, and guides further inquiry.

In this section, we present a framework[1] that enables this kind of analysis by giving a saliency score to the features of the network indicating their importance in a given context. Our framework provides unique flexibility in terms of (1) describing what behavior we would like to explain—*a prediction on a particular instance*? *A distinction between two specific classes*?—(2) drilling into highly-localized behavior or zooming out to capture global trends internalized by the model; and (3) working with the concepts learned by the model and understanding how these concepts contribute to the network's decision-making.

As the primary building block in computing this importance score, we use the gradient of the network with respect to its features. The gradient has been used in several explanation frameworks, as it provides a causal account of how a function's inputs affect its output. Intuitively, this facilitates *faithfulness*, identified in Chapter 2 as a crucial property for explanation frameworks. The sense in which this is related to a concrete notion of faithfulness is explained in Chapter 4, where we show that our framework is justified via a set of natural properties that it uniquely possesses.

[1] An implementation of our framework is available at `https://github.com/truera/trulens` [32].

3.1 Influence, Attribution, and Saliency

The majority of post-hoc explanation techniques can be considered to be what we will call *saliency measures*. A saliency measure calculates a score for each of the features of a network representing their respective importance towards the model behavior being explained. In the context of images, the saliency score of input features can easily be interpreted by simply visualizing them as a heat map localizing the important parts of the image; however, we will see that saliency scores can be used in other ways as well.

When saliency scores are to be interpreted directly (e.g., as in a heat map), the saliency measure itself constitutes an explanation framework. Recall from Chapter 2 that an explanation framework maps a model, $f : \mathbb{R}^n \to \mathbb{R}^m$, and some explanation parameters, $\phi \in \Psi$, to an explanation from some explanation space, Ω. For heat map explanations, Ω would be the set of possible heat maps, typically in the same dimension as the inputs to the model; i.e., Ω can typically be represented as $\mathbb{R}^n$.

We find it useful to make a particular distinction in the *units*, so to speak, of Ω, as they impact the way in which the explanations should be interpreted. Though this distinction is not commonly made in the literature, various proposed saliency measures have used different units. Perhaps the simplest saliency measure, first proposed by Simonyan et al. [131] in 2014, is to take the gradients of the network's output[2] with respect to its input. The resulting saliency scores are properly measured in units of output per unit of input; we will refer to saliency scores taking these units as *influence*—as in, the ability of each feature to influence the output. It has also been proposed to consider the product of the input gradients and their respective input values (i.e., "gradient times input" [2, 74, 129]). By multiplying by the input, these saliency scores measure the importance of features in terms of units of output; we will refer to saliency scores taking these units as *attribution*—as in, the portion of the output that can be attributed to each feature.

We can think of influence as a feature's *potential* to affect the model's behavior. By contrast, attribution purports to measure the realization of this potential—the absolute amount that the feature contributed to the observed outcome. In an anthropomorphic analogy, the influence of a feature could be its raw talent or skill (for producing the behavior being explained), while attribution takes into account the effort the feature dedicated to achieving the outcome. Neither the talent nor the effort of the feature alone directly imply what it will achieve.

Care must be taken when measuring and interpreting attribution. Specifically, if we would like to quantify the net impact of a feature on the model's output, this impact must be measured with respect to some baseline—*the impact relative to what*? For example, we may wish to compare the model's output on a given instance, x_1 to the output on another similar image, x_2—e.g., to understand why the model treats x_1 differently from x_2. In this case, we would measure the effort of each feature to be its deviation from its value in x_2, i.e., $x_1 - x_2$, and the difference in outputs, $f(x_1) - f(x_2)$, would be credited to the features of x_1 according to their attribution. Without a baseline, neither the effort nor the output to be accounted for are clearly defined. If we calculate attribution as the gradient times the input, we implicitly treat zero as the baseline for each feature, which may or may not be appropriate for the query we seek to address.

Additionally, when measuring attribution, it often makes sense to ensure the combined attributions of each feature exactly account for the observed output (relative to the baseline).

[2] We will clarify what we mean by taking the gradient of "the network's output" in Section 3.2

In other words, if the attribution of a feature is the portion of the baseline-adjusted output that can be attributed to that feature, then we would reasonably expect that the sum of these attributions would equal the baseline-adjusted output. This requirement corresponds to a property known in the cooperative game theory literature as *efficiency* [6, 164] (in the machine learning literature it has also been called *completeness* [140]). We discuss this point further in Section 3.4.

Both influence and attribution can be useful in different contexts. When interpreted correctly, each tells us something specific about the model our explanation framework is tasked with helping us understand. The nature of our query to the model—that is, our specific probe into the qualities of its behavior—determines which is best to use in the given context. For example, influence might tell us about potential bugs in the model's behavior that may not have been entirely manifested on a particular input. By focusing on influence we can identify the role of features which have the potential to impact the model's behavior significantly despite their degree of presence in a particular instance being relatively insignificant. On the other hand, attribution may sometimes be a better choice for understanding the fully-realized effects of existing features.

Not all saliency measures compute attribution or influence. Another family of saliency measures use modified backpropagation techniques to produce saliency scores for the inputs [7, 129, 137, 165]. Although a few of these methods can coincide with attribution measures under certain settings [74, 129], most backpropagation methods are motivated as "inverting" or "transposing" the outputs of a network back to their corresponding inputs (though not precisely in the mathematical sense of an inverse) by reversing the order of the linear transformations in the network. In this sense, the resulting saliency scores are perhaps best measured in units of input, since they map network outputs back to the network's input space. From this perspective, it is less clear how such measures ought to be interpreted. Some prior work has suggested that many backpropagation-based techniques essentially perform "(partial) image recovery [that] is unrelated to the network decisions" [108]. These findings confirm the intuition derived from a consideration of the proper units of such approaches, and indeed calls their utility into question.

3.2 Internal Influence

As discussed at the beginning of this chapter, the value in an explanation framework comes from its ability to accurately but succinctly answer a broad set of queries that add meaningfully to our understanding of a model's behavior. In this section, we present *Internal Influence*, an influence measure that serves both as an explanation framework itself (as discussed in Section 3.1), and as a key component of a broader framework of *influence-directed explanations*. Our measure is unique in its high degree of flexibility for formulating a variety of useful queries.

We developed Internal Influence using an *axiomatic approach* in order to (1) justify it as a faithful measure of influence, and (2) identify appropriate axes for generalization. By using an axiomatic justification for our framework, we avoid the need for problematic quantitative or qualitative success measures for our explanations in favor of a more principled philosophical argument; that is, one need only accept our proposed axioms to accept our influence measure (see Chapter 4 for a detailed discussion). As we will see, our axioms lead to a specific family of justified influence measures. Internal Influence encompasses this entire family of measures using a few natural parameters which capture the degrees of freedom

Figure 3.1: Two explanations for the same image with different queries. The query for the explanation on the left corresponds to, "why was the image classified as a sports car?" The query for the explanation on the right corresponds to, "why was the image classified as a sports car rather than a convertible?"

allowed by our axioms. As a result, Internal Influence generalizes other closely-related work rather than simply comprising one specific way to generate explanations; and beyond this, it explores three key axes not covered in previous work.

We refer to these axes, which form the parameter space of our influence measure, as the *quantity of interest*, the *distribution of interest*, and the *slice* of the network—they are described in detail in Sections 3.2.1-3. We subsequently put these pieces together to define Internal Influence in Section 3.2.4.

3.2.1 Quantity of Interest

The *quantity of interest* lets us specify *what* we want to explain. Up until now, we have informally referred to the quantity of interest as "the network's output," or f. If we are to be pedantic, f produces a vector in $\mathbb{R}^m$, so we must be more clear about what it means to "take the gradient of the network's output."

Often, we refer to the component of the model's output corresponding to a particular class, addressing, e.g., *why did the model classify the image in Figure 3.1 as a sports car*? Indeed, prior work has practically always used this as the de facto quantity of interest—and more specifically, the particular class has typically corresponded only to the *predicted* class, restricting the choice in quantity of interest further. While querying about the model's ultimate prediction is natural, it may sometimes be helpful to establish the evidence the model identified for *another* class that it did not end up selecting; e.g., to determine if a model acted "reasonably" even when it did not make the correct prediction.

We can also consider various combinations of outputs, allowing us to ask more specific questions, such as, *why did the model classify the image in Figure 3.1 as a sports car and not a convertible*? As shown in the figure, using the quantity of interest corresponding to "sports car" may highlight general "car features," such as tires, while a comparison of "sports car" to "convertible" might focus on the roof of the car, a "car feature" not shared by convertibles.

More generally, the quantity of interest can be represented as a function of the model's output to a scalar quantity, $q : \mathbb{R}^m \to \mathbb{R}$. In fact, when we discuss the *slice* of the network in Section 3.2.3, we will see that q need not even be a function of the network's *output* only—it may even be a function of an internal component of the network. Under this notation, it is more proper for us to say that we are explaining $q \circ f$ rather than f or "the network's output;" however, we may still informally use the latter when it is clear we are referring to the quantity of interest.

Our work is the first to consider the quantity of interest explicitly, and hence, to allow full freedom along this axis when explaining deep networks.

3.2.2 Distribution of Interest

The *distribution of interest* lets us specify the set of instances over which we want our explanations to be faithful. In our running examples thus far, we have primarily considered explanations derived for specific instances, as this is certainly the most intuitive case. More broadly, we can think of explaining the model's behavior over some distribution of points; this allows us to build an understanding of the model's more global behavior.

In an explanation framework that acts on distributions of interest (as opposed to points of interest), we can still model instance-specific inquiries by using a distribution that places all of its probability mass on a single point. Other times, we may be interested in a more general behavior over, e.g., a class of instances. Previous explanation frameworks have tended to act only on individual instances, or specific, de facto distributions tailored to a single instance.[3]

One challenge that prior work has identified with explaining neural network behavior is the relationship between the faithfulness and interpretability of explanations. It has been argued that a "fully-faithful" explanation would necessarily consist of the entire description of a model, e.g., its weights, architecture, etc. [125]. In contrast to this, we know that neural networks are not intrinsically interpretable; so clearly, effective explanations must somehow abstract from the literal model description. Hence, authors such as Selvaraju et al. [125] have viewed faithfulness and interpretability as existing within a natural trade-off.

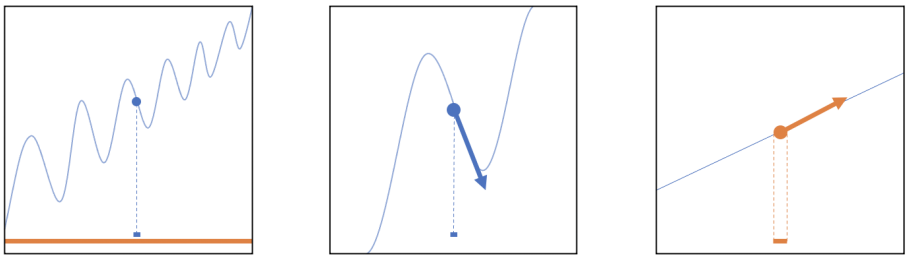
To the contrary, when the elements of a query addressed by an explanation are properly distilled, abstraction does not inherently contradict the notion of faithfulness. Perhaps the most clear case for this point is the example of explaining a single point. Our explanation may faithfully tell us about how the model behaves *on that point*, while ignoring the model's full range of behavior more globally. Namely, this explanation can be *locally faithful*, while still far less complex than the model in its entirety.

More broadly, a concept of *distributional faithfulness* can assure that explanations faithfully address the model's behavior as it pertains (in aggregate) to any distribution of points we would like to capture by our query. Figure 3.2 provides an illustration of this point. Figure 3.2a shows an example model, which corresponds to a one-dimensional function f. If we want to "zoom in" to the model's local behavior on some point, x, we can do so using a distribution of interest, D_1, that corresponds to that single point. In Figure 3.2b, for example, we see that f slopes downward on x. We can take this to mean that, in the vicinity of x, increasing x will tend to decrease the output of f.

On the other hand, if we want to "zoom out" to get a more global understanding of the model's behavior, we can use a wider distribution of interest, D_2, that captures a range of inputs to f. As shown in Figure 3.2c, the slope of f over D_2 points upwards. This can be interpreted to mean that, more generally, increasing x will tend to increase f.

In both the zoomed-in and zoomed-out case, the description of the model's behavior we receive is correct for the question being asked. Both the model's behavior on a single point and its average behavior over a region are valid to investigate, and both contribute to our understanding of the model. The confusion over how a faithful explanation framework can appropriately distill complex model behavior is clarified when we model our desired abstractions explicitly within the framework, which our work is the first to do.

[3] See Section 3.4 for discussion of de facto distributions of interest used in prior work.



(a) (b) (c)

Figure 3.2: Illustration demonstrating how the distribution of interest can help specify the level of abstraction when explaining a model's behavior. **(a)** an example model, which corresponds to a one-dimensional function f (shown in light blue). Two distributions of interest, D_1 (blue) and D_2 (orange) are shown on the x-axis, represented as bars that indicate the range of the support of the respective distribution. **(b)** zooming in to the model's behavior on D_1 (which corresponds to a singe point), we see that the gradient over this distribution, which captures the model's local behavior at this point, points downwards. **(c)** zooming out to capture the support of D_2, we see that the model's aggregate behavior over this range slopes upwards.

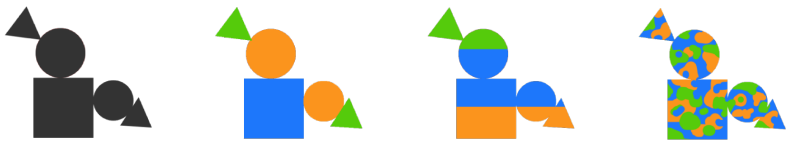
3.2.3 Internal Slice

Commonly, explanation frameworks provide *implementation invariance*; in other words, they treat the network abstractly as a single function rather than a layered composition of functions computing latent representations. At some level, this property makes sense. If two networks represent the exact same function—i.e., they produce the exact same outputs on *all* inputs—then distinguishing between their implementations may rightly be considered a strictly academic exercise. As a particularly clear example of this, consider the fact that by permuting the internal neurons of a network (and adjusting the incoming and outgoing weights accordingly), we can obtain a new network that computes the same function, but is—in a trivial sense—different from the original. It seems clear that we have no reason to desire an explanation framework to distinguish between such "isomorphic" networks.

More practically, however, it can be useful to expose the learned feature representations of a network. For example, two networks may behave the same—or sufficiently similarly—on certain observed points, but differently on other, untested points. A prescient explanation might lead us to predict this divergence in behavior without necessarily observing the points on which it is manifested.

Indeed, we have already articulated this point in a different context in Chapter 1: recall our example in Figure 1.1, where two models making the same prediction on a picture of a dog with a tennis ball did so for rather distinct reasons. Here, one model made its prediction based on the dog's face, while the other did so on account of the tennis ball. In this case, implementation invariance does not necessarily present a problem because the explanations were readily distinct without considering the nature of the models' internal representation. Presumably, the models formed some internal feature representing dogs and tennis balls, respectively, but we did not need to know how this was accomplished to discover the bug in the second model's behavior.

This need not be the case, though, so we may sometimes want to drill deeper to determine the trustworthiness of a model. Suppose, for example, in some abstract task, an explanation highlights the shape shown in Figure 3.3a as the salient part of an image. However, note



(a) (b) (c) (d)

Figure 3.3: An illustration providing a comparison of explanations that identify salient inputs indiscriminately (a) with those that distinguish salient regions according to the concepts encoded by internal neurons (b-d). **(a)** an explanation indicating the salient inputs (shown in black) for some abstract task performed by a neural network. **(b)** a possible decomposition of the salient features according to the concepts encoded by three internal neurons (represented by the colors blue, green, and yellow) which correspond to specific shapes. **(c)** alternative decomposition where the internal neurons are sensitive to the y-coordinate. **(d)** alternative decomposition where the salient inputs are distributed over the three neurons unintelligibly.

that the salient input features do not specify the model's internal view of the highlighted shape. For example, suppose the network contains three internal neurons. Internally, the model might decompose the highlighted shape between the three neurons in any number of ways, e.g., as depicted in Figures 3.3b, c, or d (where the regions of the original shape corresponding to each neuron are represented in blue, green, and yellow).

Supposing we knew that the generative process behind the data used for the task in this example was in fact decomposed as depicted in Figure 3.3b, then discovering this particular implementation detail of the network would surely inspire more trust than the explanation in Figure 3.3a alone—and certainly more than the explanations in Figures 3.3c or d. Thus, we can see how it may be desirable to peer into the internals of a deep network rather than forcing strict implementation invariance.

Explanations that reference the internals of a model can also be motivated by a desire to reason about higher-level features than the inputs of a model (e.g., raw pixels) represent. In a criticism of explanation frameworks used in health care applications, Ghassemi et al. [47] point out that localizing salient features may be inadequate in contexts like medical imaging because the locations themselves do not reveal *why* they are important. For example, even when the correct regions are identified, "[a] clinician cannot know if the model appropriately established that the presence of an airspace opacity was important in the decision, if the shapes of the heart border or left pulmonary artery were the deciding factor, or if the model had relied on an inhuman feature, such as a particular pixel value or texture that might have more to do with the image acquisition process than the underlying disease" [47]. While pixels in medical images represent far more primitive concepts than airspace opacities, etc., the learned internal features of a network can certainly—at least in principle—represent such concepts. While the meaning of the internal features cannot be known *a priori*, and thus must be ascertained in some way, this can be viewed as an orthogonal problem. More discussion on interpreting salient features is provided in Section 3.3.

To capture the ability to peer into the learned features of a network, Internal Influence uses a parameter which we call the *slice*. Informally, a slice can roughly be equated with a particular layer of a network. In a low layer, a model typically encodes more primitive features, such as edges and simple textures, while in a higher layer, the model might encode high-level features, such as dogs, tennis balls, or airspace opacities.

Put mathematically, a slice is a decomposition of a network into two functions whose composition is the function f computed by the entire network (Definition 3.1).

Definition 3.1 (Network Slice)**.** *A slice of a network, $f : \mathbb{R}^n \to \mathbb{R}^m$, is a decomposition of f into two functions, h and g, such that $f = g \circ h$.*

Essentially, the slice specifies a single window into the implementation of the model that our explanation framework is sensitive to; this leads to a relaxed version of implementation invariance that is allowed to depend on the internal layer exposed by the slice. Meanwhile, the components of the slice, g and h, are still treated as black boxes (i.e., we are agnostic to *their* implementation).

Here, g represents the "top" of the network, which performs the network's task (e.g., classification) using features that are computed by the "bottom" of the network, h. Note that slices generalize gracefully to input explanations with full implementation invariance by letting $g = f$ and setting h to the identity function. Our work is the first to extend explanations to slices beyond this degenerate case.

3.2.4 Influence Measure

We are now ready to define our influence measure, Internal Influence. Internal Influence acts on a parameter space (Ψ) that specifies the quantity of interest, q, distribution of interest, $\mathcal{D}$, and slice, (g, h). Of course, in order for (g, h) to be a valid slice of the model f, we must have that $g \circ h = f$; thus, for convenience, we will write Internal Influence, χ, as a function of $g \circ h$, q, and $\mathcal{D}$ (omitting f, which is captured by the slice).

Put succinctly, Internal influence is simply the expected gradient of the quantity of interest with respect to the output of h, taken over the distribution of interest (Definition 3.2).

Definition 3.2 (Internal Influence)**.** *Given a slice of a network, (g, h), a quantity of interest, q, and a distribution of interest, $\mathcal{D}$, internal influence, χ, is given by*

$$\chi(g \circ h,\ q,\ \mathcal{D}) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \Big[\nabla_z \big[q \circ g \big] \big(h(x) \big) \Big],$$

where the variable, z, is given by $z = h(x)$.

In theory, the expectation from Definition 3.2 is computed via the integral given by Equation 3.1, where $\mathcal{D}(x)$ is the probability mass allocated to x by the distribution of interest. Note that here, the distribution of interest is defined over the input space of f, $\mathbb{R}^n$. When it is more convenient, we can instead express the distribution of interest over the latent space induced by h on $\mathbb{R}^n$, in which case we would instead take the integral in Equation 3.1 over z.

$$\mathop{\mathbb{E}}_{x \sim \mathcal{D}} \Big[\nabla_z \big[q \circ g \big] \big(h(x) \big) \Big]_i = \int_{x \in \mathbb{R}^n} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_{h(x)} \mathcal{D}(x) \cdot dx \tag{3.1}$$

In practice, the integral in Equation 3.1 may not be feasible to compute. Instead, we can approximate it via a weighted average. In an intuitive and easy-to-implement formulation of this approach, we can consider taking the average gradient over points in a multiset, D, drawn uniformly from $\mathcal{D}$ (Equation 3.2). This formulation is particularly convenient because

it means that the distribution of interest does not have to be given explicitly; in turn, we can think of the distribution of interest as simply a collection of points.

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\Big[\nabla_z\big[q\circ g\big]\big(h(x)\big)\Big]_i \;\approx\; \frac{1}{|D|}\cdot\sum_{x\in D}\left.\frac{\partial(q\circ g)}{\partial z_i}\right|_{h(x)} \tag{3.2}$$

3.3 Interpreting Influential Features

The slice parameter allows Internal Influence to reference the internals of a model, giving insight into how it uses learned concepts. In many machine learning tasks, especially those for which deep networks are uniquely well-suited, the raw input features (e.g., RGB pixels) are not meaningful on their own. Localizing salient features does not reveal *why* those features are important. For example, a pixel may be important because of its relationship with other nearby pictures—e.g., in forming an edge—and a salient region may be important because of its color (or any other number of regions). In principle, these sorts of concepts can be represented directly by internal neurons. If we can communicate in the "language" of the network's internal representations, our explanations can directly provide insight into the role of these high-level concepts in the network's behavior.

Of course, the meaning of a neural network's internal features cannot be known *a priori*. Interpreting learned features can be viewed as a separate problem from identifying salient features, and has independently received attention in the literature [3, 73, 117, 131, 165]. In addition, Internal Influence can also serve as a tool in this effort.

Internal Influence for Interpreting Learned Features

One of the most straightforward ways to achieve a rudimentary understanding of what an internal feature represents is by visualizing the low-level inputs that contribute to its activation. For a slice of a network, (g, h), suppose we are interested in identifying the inputs contributing to the i^{th} feature in the latent space induced by h on the model's input. We can think of h as a neural network of its own, and consider the slice that captures this entire network, i.e., (h, h') where h' is the identity function. We can then construct a quantity of interest, q, that extracts the i^{th} feature from h, i.e., $q(z) = z_i$. Putting these pieces together, we can calculate the Internal Influence of the network's inputs on the i^{th} feature of h as $\chi(h \circ h', q, \mathcal{D})$.

Often, we may use this approach in a process in which we (1) determine a set of influential internal features (using Internal Influence), and (2) interpret these features by localizing them. This can be seen as separately measuring the gradient (of the network's output) that flows through a particular neuron of interest. Using this approach allows us to produce explanations that distinguish salient regions according to the concepts encoded by internal neurons like in the illustrations in Figure 3.3.

Other Interpretation Techniques

The above approach does not entirely circumvent the issue of requiring richer insight than locations of features represent. Instead, it should be viewed as a potential starting point in a scientific process, in which internal representations are examined at many angles to form a robust hypothesis of their meaning. Other methods for interpreting neurons thus play a complimentary role.

One alternative way to approach the problem is through exemplars. For example, one may visualize a collection of training images that activate a particular neuron [165]. This does not establish causation—merely correlation—but may provide further insight. In particular, exemplars provide the ability to view a variety of manifestations of a concept, and to quantify the degree to which the concept is present in various instances. For example, we may find that a particular neuron is active in images of red barns, and the strength of its activation correlates with the visual redness of the barn, activating most strongly on freshly-painted red barns. From this we might infer that the feature likely captures the barn's red color. Evidence for this could be made stronger if the activation could be localized to the barn through a faithful influence measure. This hypothesis could be further tested by providing "counterfactual" images, such as barns painted "John Deere green."

While exemplars from training (or auxiliary) data have the advantage of being in-distribution (where the model is presumably better behaved), we can also use synthetic visualizations that optimize for strongly activating a particular neuron [131]. This can give us a sense of the "essence" of what a neuron has learned, realized as its "ideal" input.

We must also consider the possibility that interpretable features, if they are learned by the network, may be manifested as linear combinations of internal features, as opposed to isolated within single neurons. Prior work has sought to establish and interpret meaningful directions in the latent space [3, 73, 117]. For example, Kim et al. [73] propose *concept activation vectors*, which capture the principle direction in a network's latent space that distinguishes between exemplars selected to exhibit a chosen concept and those selected not to.

Regardless of the method used, thorough testing is the best way to build confidence in an interpretation of the network's latent space. By challenging our hypotheses we may form more nuanced ones. Here, moving out of the distribution the network was trained on can be an asset, as it can break important structural correlations in the data that the model may or may not be sensitive to. For example, Geirhos et al. [46] studied commonly-used Imagenet models by feeding them inputs where patches of the image were permuted. In doing so, they found that the networks' performance was strikingly invariant to such permutations, suggesting that the models' features primarily reflected textures (as opposed shapes, for example).

3.4 Related Work

The problem of explaining deep networks post hoc has been studied extensively in the prior literature, *e.g.*, [7, 96, 114, 118, 125, 129, 131, 134, 140, 154, 165]. Among prior methods, Internal Influence can be sub-categorized with other gradient-based techniques, such as Saliency Maps [131], Integrated Gradients [140], and Smooth Gradients [134]. Non-gradient-based methods do not necessarily satisfy faithfulness, i.e., the features they indicate as salient to do not necessarily line up with a natural notion of causality with respect to the function computed by the network (this concept is made more mathematically rigorous in Chapter 4).

Gradient-Based Methods

As described in Section 3.2, our work generalizes prior work on gradient-based saliency measures along three key axes: (1) the quantity of interest, (2) the distribution of interest, and (3) the internal slice of the network. Here, (1) and (2) provide greater freedom to design

appropriate queries for understanding key network behaviors, and (3) allows us to reason on the level of the concepts encoded by the network.

As Internal Influence is more general than prior gradient-based approaches, these other methods can naturally be realized as specific parameterizations of Internal Influence. In particular, while most methods have not considered slices or general quantities of interest, they essentially differ by establishing a particular family of distributions of interest. We outline three key such cases below.

Saliency Maps. Saliency Maps, proposed by Simonyan et al. [131], are perhaps the simplest gradient-based influence measure, computed as simply the input gradient at a single point being explained. Naturally, this corresponds to a distribution of interest in which the entire probability mass is placed on the single point in question. As such, Saliency Maps constitute the most local view of the network's behavior.

Smooth Gradients. Another measure, Smooth Gradients, proposed by Smilkov et al. [134], offers a wider view of the network's behavior by averaging the gradients taken at points sampled from a Gaussian distribution centered at the point being explained. This is modeled in Internal Influence by using a Gaussian distribution of interest.

Integrated Gradients. Sundararajan et al. [140] proposed Integrated Gradients, a gradient-based approach for measuring attribution that possesses several key properties. Integrated Gradients is defined as the path integral of the gradients along a straight line segment connecting a baseline, x_0, to the point being explained. This can be written as Equation 3.3.

$$\chi^{IG}(f, x, x_0) = (x - x_0) \int_{\alpha=0}^{1} \left.\frac{\partial f}{\partial x}\right|_{x_0+\alpha(x-x_0)} d\alpha \tag{3.3}$$

Integrated Gradients satisfies *efficiency*, a property for attribution measures discussed in Section 3.1 which states that $\sum_i \chi_i^{IG}(f, x, x_0) = f(x) - f(x_0)$, i.e., that the input attributions account exactly for the total change in output of the function from the baseline to x.

The integral in Equation 3.3 corresponds to a measure of Internal Influence where the distribution of interest is given by a uniform distribution over the line segment between x and x_0. Despite the fact that Integrated Gradients measures *attribution* while Internal Influence measures *influence*, we see from the above observation that the latter can be converted to the former by multiplying by $(x - x_0)$, the "effort" of the features relative to the baseline. Here, this conversion is "valid" because it achieves *efficiency*, as stated earlier.

Backpropagation Methods

Zeiler and Fergus [165] proposed Deconvolution as one of the first methods for probing the behavior of deep networks. Essentially, Deconvolution propagates in reverse from the network's output to its input to trace the output activations back to corresponding input activations. Similar styles of techniques with various refinements have followed this work [7, 129, 137]. More recently, some of these approaches have come under question as they have been found to essentially perform partial image recovery [108], and to be concerningly invariant to the parameters of the network [2].

Others can be parameterized to compute a form of attribution (input times gradient) [74, 129]; however, unless they are parameterized to be equivalent to Integrated Gradients, they may not achieve efficiency.

Class Activation Mapping

Another popular family of explanation approaches, not necessarily mutually exclusive with the types of approaches discussed so far, is derived from a technique called *class activation mapping* (CAM) [172]—several variants of this approach have been proposed [21, 125, 151]. In essence, these methods localize salient regions in images by up-scaling the salient regions in the latent space of the network (which has lower resolution due to down-sampling operations in the network, such as pooling). While in a sense, these methods can be thought of as computing a form of internal salience, a few points bear mentioning.

First, the saliency in the latent feature maps may be computed in a number of ways. While Grad-CAM [125] uses the gradient (an instance of Internal Influence), several other CAM approaches use different methods, e.g., backpropagation. Even when Internal Influence is used, some granularity is lost, as the feature influences are aggregated over the channels, which represent distinct features. As a result, only the locations (in latent space) of features are preserved, while the further decomposition into learned concepts is not.

Second, the extrapolation of the latent space to the input space via up-scaling is somewhat suspect. While the limited receptive field of internal neurons partially constrains the locations of the corresponding inputs, naive up-scaling is likely too simple to adequately capture this relationship. This low-resolution solution leaves the resulting heat maps more open to interpretation, which opens the door to confirmation bias.

Chapter 4

Assessing the Quality of Explanations

Many possible methods for explaining neural network behavior, including our own, have been proposed in what has become a vast body of literature [7, 21, 41, 85, 114, 125, 129, 131, 137, 140, 151, 154, 165]. The question naturally arises, *how do we know which explanation frameworks are best*? More broadly, *what constitutes a good explanation*? Or perhaps more pointedly, *which explanation frameworks can we trust*?

One core function of an explanation framework is to asses the conceptual soundness of the model being explained. As such, we should anticipate that the quality of the model will impact the explanations we will see. We are therefore faced with the challenge of being able to discern between bad explanations for good models and good explanations for bad models, as either may be unintelligible, counter-intuitive, or otherwise apparently undesirable. Conversely, while good explanations for conceptually sound models should add clarity and instill trust, we want to avoid being led astray by bad explanations for conceptually unsound models that offer a false sense of security.

We argue that, in light of this observation, explanations cannot be evaluated or warranted purely on the basis of their appearance; rather, an explanation framework should be justified from first principles, which do not require *a priori* knowledge of the quality of the underlying model. If an explanation framework is justified in this manner, there is no need to empirically evaluate its results, provided it can be demonstrated to be sufficiently expressive (in the sense given by Definition 2.2 in Chapter 2). The quality of said explanations can instead be confidently interpreted as a reflection of the underlying model's conceptual soundness *with respect to the query the explanation addresses*.

Note that this still requires that we ask the right questions, and that our framework supports asking those questions—which is why the flexibility of Internal Influence is instrumental. Without the right questions, our explanations may still fail to extract useful insights concealed behind opaque computations.

However, beyond its flexibility, we demonstrate in Section 4.1 that Internal Influence is justified via an essential set of axioms—properties that are natural to expect from any measure of feature influence within a neural network. Moreover, we show that the family of influence measures captured by Internal Influence are *the only* measures satisfying these axioms, establishing Internal Influence as uniquely justified for influence-directed explanations.

Any arbitrary decisions for measuring influence are captured succinctly by the parameters of Internal Influence: the slice, and the quantity and distribution of interest.

The choice of using an axiomatic justification is relatively rare in the literature on explanations. In Sections 4.2 and 4.3 we discuss (and critique) a variety of other approaches that have been used for assessing explanation quality. These methods can be broadly categorized as either (1) empirical tests that serve as de-facto axioms (discussed in Section 4.2) or (2) tests mistakenly applied to explanations that should be better understood as desiderata of the underlying model (discussed in Section 4.3).

4.1 Axiomatic Justification of Internal Influence

In this section, we justify the family of measures captured by Internal Influence (Definition 3.2 in Chapter 3) by defining a set of natural axioms for influence measures, and then proving a tight characterization. We first address the the case where the influence is measured with respect to inputs—i.e., when the slice, (g, h), is given as $g = f$ and $h(x) = x$. and then generalize to general slices that capture internal layers. This approach is inspired, in part, by axiomatically justified *power indices* in cooperative game theory [6, 164], which have inspired a few other explanation frameworks as well [31, 140]. An important difference between our work and similar aforementioned axiomatizations, as we elaborate below, is that we carefully account for distributional faithfulness in this work.

Input Influence

We begin by characterizing an ideal measure, $\chi^{\mathsf{input}}(f, d, \mathcal{D})$, that measures the influence of the features of a model, f, on a quantity of interest, q, over a distribution of interest, $\mathcal{D}$. In other words, we will consider a simplification of the parameter space of Internal Influence, wherein we will ignore nontrivial slices of the network for the moment.

The simplest neural network we can consider calculating influence for is a linear model, i.e., where there are no hidden layers, and the logits of the network are given simply by $f(x) = xW + b$. The interpretation of a linear model is straightforward since a unit change in an input corresponds to a change in the output given by the coefficient, independent of the other feature values or the input under consideration. Verifying that the feature influences align with this straightforward intuition on the simplest of models serves as a basic sanity check for any influence measure. Accordingly, the first axiom, *linear agreement* (Axiom 4.1), states that for linear models, the coefficient of an input is its influence. Axiom 4.1 is formulated in terms of the quantity of interest (composed with the model function), $q \circ f$, since this is the actual function that we measure influence on. This formulation can principally be taken to mean that the axiom refers to simple quantities of interest that select a scalar output from a linear model of the form $f(x) = xW + b$ (though in reality it means something slightly more general).

Axiom 4.1 (Linear Agreement)**.** *For linear quantities of interest of the form*

$$(q \circ f)(x) = w^T x + b = b + \sum_i w_i \cdot x_i,$$

the influence of feature x_i is given by $\chi_i^{\mathsf{input}}(f, q, \mathcal{D}) = w_i$.

The second axiom, *distributional marginality* (Axiom 4.2), is inspired by the *marginality principle* [164] in prior work on cooperative game theory. The marginality principle states in essence that an input's importance only depends on its own contribution to the output. Formally, if the partial derivatives with respect to an input of two functions are identical at all input instances, then that input is equally important for both functions.

Axiom 4.2 is a weaker form of this requirement that only requires equality of importance when partial derivatives are same for points in the support of the distribution. This axiom ensures that the influence measure only depends on the behavior of the model on points within the manifold containing the input distribution. This property captures the notion of *distributional faithfulness* described informally in Section 3.2.2; namely, that while an explanation must accurately describe a model's behavior, we can still simplify the information conveyed by explanations that are faithful in this sense by tailoring them to describe local behavior.

Axiom 4.2 (Distributional Marginality)**.** *Let $X \in \mathbb{R}^n$ be a random variable distributed according to the distribution of interest, $\mathcal{D}$. If for neural networks, f_1 and f_2, and quantities of interest, q_1 and q_2,*

$$\Pr\left[\left.\frac{\partial(q_1 \circ f_1)}{\partial x_i}\right|_X = \left.\frac{\partial(q_2 \circ f_2)}{\partial x_i}\right|_X \right] = 1,$$

then $\chi_i^{\mathsf{input}}(f_1, q_1, \mathcal{D}) = \chi_i^{\mathsf{input}}(f_2, q_2, \mathcal{D})$.

Finally, the third axiom, *distributional linearity* (Axiom 4.3), states that the influence measure is linear in the distribution of interest. This ensures that influence measures are properly weighted over the input space, i.e., influence on infrequent regions of the input space receive lesser weight in the influence measure as compared to more frequent regions. Practically, this admits the most natural interpretation of the distribution of interest, namely, that points contributing to the influence score are weighted according to their probability mass. To formalize this axiom, we consider distributions of interest that are expressible as essentially a "weighted combination" of some family of distributions, $\mathcal{P}$; Axiom 4.3 stipulates that the influence over such distributions of interest can be likewise expressed as a "weighted combination" of the influences calculated over the members of $\mathcal{P}$.

Axiom 4.3 (Distributional Linearity)**.** *Let $\mathcal{P}$ be family of distributions indexed by $a \in \mathcal{A}$. If a distribution of interest, $\mathcal{D}$, can be written in terms of the distributions $P_a \in \mathcal{P}$, and weight function $\alpha(a)$, as*

$$\mathcal{D}(x) = \int_{a \in \mathcal{A}} \alpha(a) \cdot P_a(x) \cdot da,$$

then

$$\chi^{\mathsf{input}}(f, q, \mathcal{D}) = \int_{a \in \mathcal{A}} \alpha(a) \cdot \chi^{\mathsf{input}}(f, q, P_a) \cdot da.$$

The requirements imposed by these three axioms appear rather basic. Essentially, we have argued that a reasonable influence measure should (1) be consistent with the straightforward interpretation of linear models, (2) reflect the marginal impact of individual features, and (3) admit the natural interpretation of the distribution of interest as weighting the degree to which the various parts of a model's input space should contribute to the influence score. Interestingly, even with these minimal requirements, we can show that the only influence

measure that satisfies these three axioms is the expected input gradient over the distribution of interest (Theorem 4.1). In other words, these three axioms uniquely characterize the family of influence measures that are equivalent to some instantiation of Internal Influence tailored to input-output behavior.

Theorem 4.1. *The only measure that satisfies linear agreement, distributional marginality and distribution linearity is given by*

$$\chi_i^{\mathsf{input}}(f, q, \mathcal{D}) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \Big[\nabla_x \big[q \circ g \big](x) \Big]_i = \int_{x \in \mathbb{R}^n} \left. \frac{\partial (q \circ f)}{\partial x_i} \right|_x \mathcal{D}(x) \cdot dx$$

Proof. Assume some influence measure, χ^{input}, satisfies Axioms 4.1-3.

Choose any differentiable function $q \circ f$, and let $P_a(x) = \delta(x - a)$, where δ is the Dirac delta function—that is, P_a places all its probability mass on the single point a. Let f' be a linear function whose slope is given by the gradient of $q \circ f$ with respect to x evaluated at a (Equation 4.1), and let q' be the identity function.

$$f'(x) = \Big(\nabla_x \big[q \circ f \big](a) \Big)^T x \tag{4.1}$$

By Axiom 4.1, it must be the case that $\chi^{\mathsf{input}}(f', q', P_a)$ is given by Equation 4.2—indeed, this holds even independently of the particular choice of distribution of interest.

$$\chi_i^{\mathsf{input}}(f', q', P_a) = \left. \frac{\partial (q \circ f)}{\partial x_i} \right|_a \tag{4.2}$$

By Axiom 4.2, we thus have that $\chi^{\mathsf{input}}(f, q, P_a) = \chi^{\mathsf{input}}(f', q', P_a)$, since on P_a, whose support is only the point a, the gradients of $f \circ q$ and $f' \circ q'$ align.

Any distribution, $\mathcal{D}$, over $\mathbb{R}^n$ can be written according to Equation 4.3, owing in part to the fact that $P_a(x) = 0$ unless $a = x$.

$$\mathcal{D}(x) = \int_{a \in \mathbb{R}^n} \mathcal{D}(a) \cdot P_a(x) \cdot da \tag{4.3}$$

Therefore, by Axiom 4.3, we obtain Equation 4.4.

$$\chi_i^{\mathsf{input}}(f, q, \mathcal{D}) = \int_{a \in \mathbb{R}^n} \mathcal{D}(a) \cdot \chi_i^{\mathsf{input}}(f, q, P_a) \cdot da = \int_{x \in \mathbb{R}^n} \left. \frac{\partial (q \circ f)}{\partial x_i} \right|_x \mathcal{D}(x) \cdot dx \tag{4.4}$$

Finally, we remark that it is not difficult to verify that χ^{input} as defined in Theorem 4.1 satisfies Axioms 4.1, 4.2, and 4.3. □

Internal Influence

We now show how Internal Influence can be derived by generalizing the characteristic measure of input influence given in Theorem 4.1 to one that can be used to measure the influence of internal neurons as well. We do so by introducing an additional axiom, *preprocessing* (Axiom 4.4), which essentially encodes a consistency requirement equating the influence of an internal neuron, z_j, to the ideal input influence (i.e., according to Theorem 4.1) of an input

that is distributed exactly the same as z_j. That is, suppose we have a network, f_1, that can be sliced as $g \circ h$, and suppose further that we have another network f_2 where $f_2 = g$. Applying h to a distribution of interest $\mathcal{D}_{\mathcal{X}}$ over the inputs of f_1 induces a new distribution, $\mathcal{D}_{\mathcal{Z}}$, over the inputs to f_2. We can thus consider the input influence for the features of f_1 with the distribution of interest given by $\mathcal{D}_{\mathcal{Z}}$. Naturally, this influence measure should not be affected by whether $\mathcal{D}_{\mathcal{Z}}$ was imagined *a priori*, or if it arose from applying a section of some neural network—e.g., h—to another distribution.

This argument is especially clear if we consider the case where h is the identity function. In this case, $f_1 = f_2$, and the inputs to f_2 are exactly the same as the internal features determined by the slice (g, h), making it difficult to justify how their influences could be different.

Axiom 4.4 (Preprocessing)**.** *Let (g, h) be a slice of a neural network, f. Given distribution $\mathcal{D}_{\mathcal{X}}$ over the input space of f, let $\mathcal{D}_{\mathcal{Z}}$ be the probability distribution over the input space of g, induced by applying h to $x \sim \mathcal{D}_{\mathcal{X}}$, given by*

$$\mathcal{D}_{\mathcal{Z}}(z) = \int_{x \in \mathbb{R}^n} \mathcal{D}_{\mathcal{X}}(x) \cdot \delta(h(x) - z) \cdot dx.$$

Then, $\chi^{\textit{input}}(g, q, \mathcal{D}_{\mathcal{Z}}) = \chi(g \circ h, q, \mathcal{D}_{\mathcal{X}})$.

We now demonstrate that the addition of Axiom 4.4 uniquely characterizes Internal Influence, as defined by Definition 3.2, or equivalently, Equation 3.1. This result is stated in Theorem 4.2; we note that while Theorem 4.2 only references preprocessing (Axiom 4.4), Axioms 4.1-3 are implicitly invoked through this axiom.

Theorem 4.2. *The only measure that satisfies preprocessing is Internal Influence (Definition 3.2), given by*

$$\chi(g \circ h, q, \mathcal{D}) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \Big[\nabla_z \big[q \circ g \big] \big(h(x) \big) \Big]_i = \int_{x \in \mathbb{R}^n} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_{h(x)} \mathcal{D}(x) \cdot dx.$$

Proof. Assume some influence measure, χ satisfies Axiom 4.4. Let (g, h) be a slice of a neural network, f, let q be a quantity of interest, and let $\mathcal{D}_{\mathcal{X}}$ be a distribution of interest over the domain of f. Let $\mathcal{D}_{\mathcal{Z}}$ be the probability distribution over the input space of g, induced by applying h to $x \sim \mathcal{D}_{\mathcal{X}}$, as defined by Equation 4.5

$$\mathcal{D}_{\mathcal{Z}}(z) = \int_{x \in \mathbb{R}^n} \mathcal{D}_{\mathcal{X}}(x) \cdot \delta(h(x) - z) \cdot dx \tag{4.5}$$

By Axiom 4.4, we have Equation 4.6 for any index, $i \in [k]$ (where k is the input dimension to g). By applying the definition of $\chi^{\textsf{input}}$, we obtain (4.7). By assumption, we can replace $\mathcal{D}_{\mathcal{Z}}$ to obtain (4.8), which can be rearranged to (4.9). As a property of the Dirac delta function, the integral over z can be simplified to obtain (4.10).

$$\chi_i(g \circ h, q, \mathcal{D}_{\mathcal{X}}) = \chi_i^{\mathsf{input}}(g, q, \mathcal{D}_{\mathcal{Z}}) \tag{4.6}$$

$$= \int_{z \in \mathbb{R}^k} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_z \mathcal{D}_{\mathcal{Z}}(z) \cdot dz \tag{4.7}$$

$$= \int_{z \in \mathbb{R}^k} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_z \int_{x \in \mathbb{R}^n} \mathcal{D}_{\mathcal{X}}(x) \cdot \delta(h(x) - z) \cdot dx \cdot dz \tag{4.8}$$

$$= \int_{x \in \mathbb{R}^n} \mathcal{D}_{\mathcal{X}}(x) \int_{z \in \mathbb{R}^k} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_z \delta(h(x) - z) \cdot dz \cdot dx \tag{4.9}$$

$$= \int_{x \in \mathbb{R}^n} \left. \frac{\partial (q \circ g)}{\partial z_i} \right|_{h(x)} \mathcal{D}_{\mathcal{X}}(x) \cdot dx \tag{4.10}$$

Finally, it is clear from the equivalence of Equations 4.6-10 that Internal Influence as defined previously satisfies Axiom 4.4. □

4.2 Implicit Axioms via Empirical Tests

A general dissatisfaction with the appearance of explanations produced by existing methods, perhaps, has led to a large volume of work that attempts to explore novel ways of explaining complicated network behavior. A choice among a multitude of methods must be justified in some way or another. Rather than taking an axiomatic approach, as we have advocated, much of the existing literature follows a more empirical approach to evaluate various explanation frameworks, using quantitative (or even qualitative) tests to measure explanation quality. Within the crowded space of work on explaining neural network behavior, various explanation approaches may distinguish themselves on particular empirical tests. Of course, this raises the question of which tests are themselves justified.

Most empirical tests used to evaluate explanation frameworks can be broadly separated into two categories, differentiated by whether they are evaluated on the basis of the model and explanation parameters alone, or if they require access to some auxiliary information related to an explanation criterion (Ω^*). In this section we will consider the former; Section 4.3 will provide discussion of the latter.

We will argue that empirical tests that can be evaluated solely on the basis of inputs that would be available to an explanation framework—to the extent that they are considered the most important measure of explanation quality—in fact constitute *implicit axioms*. By stating axioms implicitly in the form of a test, their use as a justification may evade due scrutiny, as the implications of such axioms and their interactions with other desirable properties of explanation frameworks become less apparent. If we are to accept such tests as justified, we should seek an analysis of what they entail as explicit axioms.

One of the most common empirical tests in this category, which we will refer to as the *occlusion test*, is to measure the drop in logit or confidence output of the network f as the parts labeled as salient by the explanation are occluded in some fashion [7, 21, 122]. Several different precise methods for taking this measurement have been proposed. First, the method of occlusion may vary, from blacking- or graying-out pixels (or replacing them with some baseline value), to blurring or adding noise to salient regions. Second is the question of how

to score the drop in f's output. Aside from the decision of whether to use the unscaled logit outputs, or the scaled confidence scores, one may focus on the drop induced by occluding the single-most salient pixel or region, the area under the curve induced by plotting f as regions are occluded, the number of regions that must be occluded to reach a particular drop, etc.

As an aside, let us consider how gradient-based explanation methods, which we have justified via the marginality principle and linear agreement, fare under the such an evaluation. The simplest gradient-based method is to simply take the input gradient at the single point to which the explanation applies. This is often referred to in the literature as *saliency maps* or *sensitivity analysis* [131], and corresponds to the simple case in Internal Influence when the distribution of interest is taken to be a single point. Samek et al. [122] find that saliency maps do not perform well under the occlusion test. If we consider what the input gradient means—the direction in the input space leading to the steepest ascent for f—there is an extent to which this intuitively lines up with the goal of the occlusion test. However, as even Samek et al. point out, gradients identify a "very particular direction for reducing the prediction," while occlusion seeks regions that *independently* degrade the prediction. In other words, gradients capture features that act in concert, while occlusion disregards correlations between features.

Moreover, the gradient is highly localized in the sense that it measures the impact of *marginal* changes to the model's inputs. Meanwhile, depending on the method of occlusion, we may be introducing large changes that cannot be extrapolated from a local gradient, especially if the surface of f is highly erratic (a potential sign of conceptual unsoundness). If unstable local behavior concealing unwanted behavior [48, 141] concerns us, the fine-grain resolution of point-gradients (compared to occlusion) may in fact be desirable. However, if we are interested in a higher-level description of the model's behavior than point-gradients allow, Internal Influence allows us to widen the scope of the gradient computation via the distribution of interest. For example, a Gaussian distribution about the point being explained could be used to aggregate surrounding gradient information. Alternatively, the distribution of interest could be used to model specific occlusion types, for example by aggregating the gradients obtained as pixels are gradually blacked-out (to model black-out occlusion).

Finally, gradients are perhaps not treated fairly by the occlusion test in the sense that a positive gradient for a feature, x_i, indicates that f will increase if x_i increases, and vice versa. If we assume that occlusion will decrease x_i's value, then a large positive gradient indeed suggests that occluding x_i will reduce f. However, this assumption need not hold. If, for example, x_i is a black pixel and our method of occluding x_i is to replace it with gray (such that our occlusion is properly centered), we are directly opposing the intuition of the occlusion method under the correct prediction of the gradient. For this reason, the value of each feature relative to its occluded value should be taken into account—at the very least in determining the sign of the gradient used for the occlusion test. Put differently, while gradients measure *influence*, *attribution* (see Chapter 3 for this distinction) is the appropriate measurement to consider for the occlusion test.

Indeed, Integrated Gradients [140], does precisely this. The feature values relative to a baseline (in this case, the value taken under occlusion) are taken into account to produce a measure of total feature attribution. In fact, when using (baseline-adjusted) attribution along with the particular distribution of interest used by Integrated Gradients (discussed in Chapter 3), a property known as *efficiency*, or *completeness*, is obtained; this entails directly that the attribution of each feature is exactly the value that f would take if that feature were occluded. Thus for the initial occlusion, a careful choice of distribution and

attribution adjustment tailors gradients perfectly for the occlusion test, although this does not necessarily extend to subsequently-occluded features.

Another empirical measure of explanation quality that has appeared in the literature is a natural counterpart to the occlusion test, which we will refer to as the *isolation test*. This test occludes all but the most salient parts of the image with the hopes of increasing the network's confidence in its prediction [21]. This method is perhaps not as well-motivated as the occlusion test; while the context in the less-salient parts of the image may sometimes be distracting, it is not clear that the network would necessarily be more confident on points that deviate from the distribution on which the model was trained (the isolated features would surely be out-of-distribution).

When tests such as the occlusion test and the isolation test are employed in the literature, they typically evaluate a variety of explanation frameworks that are derived *independently* of the test criteria. This raises natural questions regarding the apparent mismatch between the way explanation frameworks are constructed and the way they are evaluated. If the test is truly the best measure of explanation quality, one ought not to compute explanations through roundabout means; we can construct the explanation framework to optimize for our intended test directly. For example, if we took the occlusion test at face value, we could easily define an explanation framework that ranks the features sequentially by greedily selecting the unranked feature whose occlusion leads to the greatest drop in confidence in the model's prediction.

More broadly, any test, $q : (\mathbb{R}^n \to \mathbb{R}^m) \times \Psi \to \Omega \to \mathbb{R}$, which, given a model, f, and parameters, $\phi \in \Psi$, produces a real-valued score for any candidate explanation, can be translated into an explanation framework, χ^*, that is optimal with respect to q. This can be done by simply defining χ^* to produce the explanationn that receives the best score from q (Equation 4.11). Provided sufficient structure in q and Ω, χ^* may be efficient to compute. Otherwise χ^* can be approximated, resulting in an approximately optimal "best-effort" explanation framework; but note that even in this case, we would explicitly aim to maximize q rather than taking an indirect approach.

$$\chi^*(f\ ;\ \phi) := \operatorname*{argmax}_{\omega \in \Omega} \big\{ q(f, \phi\ ;\ \omega) \big\} \tag{4.11}$$

The framework χ^* would be justified as the "correct" and "ideal" method for calculating explanations with respect to the test, q, in question. Thus, we can think of q as an axiomatic justification for χ^*. It is in this sense that such tests can be considered implicit axioms.

While it has perhaps not been presented in this light before, many explanation frameworks have in fact used a similar concept to the occlusion test as part of their definition [41, 114, 129, 151]. However, because this has not been made explicit, there has not been a thorough analysis (like the one in Section 4.1) of the implications of possessing an "occlusion-optimal" property.

4.3 Desiderata of the Underlying Model

While there are disadvantages to lacking an explicit characterization of the key properties of an explanation framework, there is not an inherent problem to using tests that act implicitly as axioms, provided the tests themselves are well-justified. On the other hand, another class of empirical tests have often been used in the literature that retain a more fundamental flaw. Specifically, these tests make the mistake of conflating desiderata of the explanation framework with desiderata of the model itself. Accordingly, they evaluate explanations under the

a priori assumption that the underlying model is conceptually sound. Clearly, this assumption need not hold—in fact, testing this assumption is a core function of an explanation framework in the fist place.

Perhaps the most common empirical test of this sort is what we will call the *localization test*. Many variants of this test have been used in the literature [21, 41, 151, 154, 170], but all variants of the test essentially measure the alignment between the salient pixels of an image (according to the explanation framework) and human-supplied bounding boxes that localize the subject of the image corresponding to its label. This test is tempting because it forms a reasonable proxy for conceptual soundness that can be quantitatively measured using annotations that already exist on many popular benchmark datasets. Of course, the localization test fails as an evaluation of an explanation framework because it forgets that it is not incumbent on the *explanation framework*—but rather the *underlying model*—to produce conceptual soundness.

A few prior works have also designed human subject studies as a way to assess the quality of explanations. For example, Chattopadhay et al. [21] conducted an experiment in which their explanation framework was compared to a another framework by human test subjects. In the experiment, explanations for several instances were generated using two separate methods on a single model. The explanations were then shown to 13 subjects (inexperienced in machine learning), who were asked to select the explanation that invoked the most trust in the underlying model. The crucial flaw in this experiment is its failure to initially establish the degree of trust that *ought* to be instilled in the model in question. Though it may have been unknown to the subjects, the two explanations they were presented with corresponded to *the same model*. It is not at all clear that the explanation instilling more trust between the two was the explanation instilling the *correct* amount of trust. Moreover, if the underlying model was not conceptually sound (not an unlikely prospect), then it may indeed have been preferable for the explanation to instill *less* trust in the model. While Chattopadhay et al. used the fact that subjects had preferred their explanations to claim theirs the superior explanation framework, we can see that no such conclusion can be made.

In a slightly less flawed human-based experiment, Selvaraju et al. [125] trained two separate models which attained different validation accuracies. Explanations were generated on several instances using a single explanation framework on each of the two models. The instances were controlled such that both models made the correct prediction on every instance used in the experiment. Test subjects were then asked to use the explanations to select which of the two models they trusted more. In this experiment, Selvaraju et al. were able to establish a correlation between user trust and model performance; i.e., the users tended to prefer the explanations produced on the model with the higher validation accuracy. Compared to the experiment of Chattopadhay et al., Selvaraju et al. at least take into account the fact that the quality of the explanation must be tied to the conceptual soundness of the model. Taking accuracy as a proxy for conceptual soundness, this experiment is properly formulated; if the explanations reflect the quality of the model (measured here as the validation accuracy), then they can be considered useful rather than misleading. However, while we should expect a model that is fully conceptually sound to perform well, it is not necessarily correct to assume that the more accurate model is more conceptually sound. As we will explore in greater detail in Parts II and III, even conceptually *unsound* features can lead to good performance, though such models would not hold up to distribution shifts or adversarial manipulation.

In fairness, most works that use this flavor of empirical test sell them as "evaluating

human trust," which is properly distinguished from evaluating faithfulness. To consider these concepts separately is not entirely misled. For example, a model is always a faithful explanation of itself, but this is entirely useless for providing any additional insight into its behavior. Thus, a useful explanation must provide some abstraction over the original model, meaning that there may indeed be variation in the extent to which two faithful explanations convey illuminating information.

However, when faithfulness is not taken seriously enough, falling out of favor in lieu of better "interpretability," the results may be misleadingly appealing. For example, Adebayo et al. [2] found that several explanation frameworks that produced visually pleasing explanations failed basic sanity checks for faithfulness. In particular, methods based on guided backpropagation through the network [125, 137] (these methods perform a similar calculation to backpropagating gradients, but with modifications that "clean" the propagated values, leading to more refined visualizations) were found to be concerningly insensitive to the model. Specifically, the visual appeal of the explanations produced by these frameworks persisted even when the network weights were randomized, indicating the interpretability of such explanations was not derived from the model. By contrast, similar experiments did not raise concerns over gradient-based methods, which we have seen can be axiomatically justified.

Explanations that achieve interpretability independently of the underlying model must be seen as suspect; the findings of Adebayo et al. warn us that we should be cautious of relying on interpretability measures, which do not indicate *how* interpretability is achieved.

Other methods besides guided backpropagation may dubiously achieve interpretability as well. For instance, Ribeiro et al. [118] proposed *Local Interpretable Model-agnostic Explanation* (LIME), a framework that explains the behavior of a complex model through an interpretable model (e.g., a linear classifier) that is locally faithful to the underlying model's behavior—on a collection of high-level "interpretable representations"—at a given point. While the authors intend the term "model-agnostic" (from which LIME gets its name) to mean that their approach is black-box, their concept of atomic interpretable representations introduces a problematic source of infidelity to the model being explained.

In the context of images, LIME creates an interpretable proxy model that operates on so-called "super-pixels" rather than the raw features (RGB pixels). The super-pixels used by LIME are defined generally, but in practice are obtained via standard image segmentation algorithms. These methods for segmentation are oblivious to the concepts learned by the model, and by contrast, are designed to appeal to rudimentary human visual intuition. As such, LIME presents the model as using reasonable concepts (leading to pleasing explanations), without any regard to the model's true internalized concepts.

In general, classical computer vision algorithms, such as edge detection, could be used as a filter over instances that would appear as an intuitive explanation [2]. Of course, we should not accept such explanations, because they are not a function of the model being explained.

Another slightly different place in which we must take care when interpreting explanations is when multiplying a measure of influence by the corresponding neural activations to obtain attributions. Integrated Gradients, for instance, does this. If influence (especially averaged over the distribution interest implicitly specified by Integrated Gradients) is to be interpreted as a *potential* impact a feature may have, by taking its activation into account we are essentially measuring the degree to which the potential has been realized. However, in practice this means that—if we use a black image as the baseline, as is commonly done [140]—lighter pixels will tend to have higher attribution. While this is not strictly

incorrect, it signals that we must take care in interpreting such results (the correct interpretation of Integrated Gradients is clear given its axiomatization), and moreover, in how we formulate our queries to the model. For example, in an image with a black background and a light subject, using Integrated Gradients with a black baseline essentially constitutes a "loaded question."

Conclusions. As Samek et al. [122] aptly put, "one should keep in mind that [an explanation] always represents the classifier's view, i.e., explanations neither need to match human intuition nor focus on the object of interest." Indeed, we cannot assume *a priori* that our models are conceptually sound, even when they appear proficient at the task they were trained for—vulnerabilities may we lurking still. Although we want our explanations to effectively distill sound behavior when it is to be found hidden beneath opaque computations, we must consider the possibility that unintelligible, counter-intuitive, or otherwise undesirable explanations may simply be a reflection of a neural network's lack of conceptual soundness. Put concisely, *quality explanations require quality models.*

Part II

Training Robust Neural Networks

Chapter 5

Adversarial Examples and Provable Robustness

In Part I we concluded that the quality of a model is to blame for faithful explanations that are unintelligible, counter-intuitive, or otherwise undesirable. Much of the continued effort on developing new explanation techniques—as well as some of the faulty ways of evaluating them, described in Chapter 4—has been inspired by a general dissatisfaction with the appearance of explanations produced on typical models. In this chapter, we will demonstrate that vulnerability to adversarial input perturbations (known as *adversarial examples*), a ubiquitous problem in modern neural networks, is a major factor limiting the intuitiveness of explanations. The presence of adversarial examples establishes a model as conceptually unsound, rendering qualitative evaluations of explanations are meaningless if unresolved. This will serve as a significant part of the motivation for the remainder of Part II of this thesis, which focuses on defenses against adversarial examples—though in addition to compromising conceptual soundness, adversarial examples also pose a direct security threat to models that are vulnerable to them.

To this end, we introduce *local robustness*, a property that ensures the protection of specific points against certain kinds of adversarial manipulations. Local robustness is a key property behind the wider goal of models that are invulnerable, or at least resistant, to adversarial examples, informally referred to as "robust models." This chapter contributes a possible formalization of robust models that raises the notion of local robustness—which applies to specific *points*—to *models*.

One of the primary goals of this chapter is to make explicit the vital link between the robustness of a model and its conceptual soundness. While many of these connections come from prior work, which has documented the interpretability benefits of robustness well, we take this logic a step further, arguing that robustness is a necessary *prerequisite* for conceptual soundness.

5.1 Conceptual Soundness and Adversarial Examples

The existence of what have been termed *adversarial examples* has been reported on almost as early as the onset of deep networks' surge in popularity. In 2014, Szegedy et al. [141] observed that "[by] applying an imperceptible non-random perturbation to a test image, it

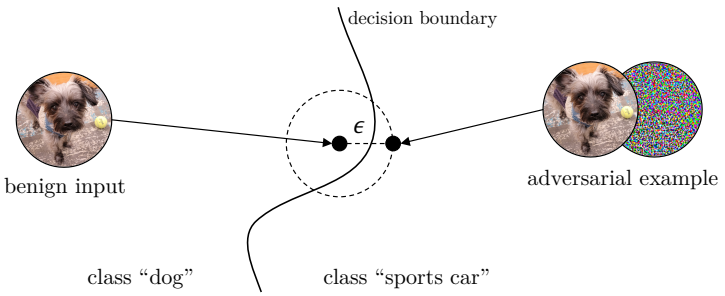


Figure 5.1: Geometric interpretation of small-norm adversarial examples.

is possible to arbitrarily change the network's prediction." A hypothetical example of this phenomenon was presented in Chapter 1 in Figure 1.2, in which an image of a dog was perturbed to lead a model to predict the class "sports car."

The concept of an adversarial example is actually quite general; although certain classes of adversarial examples can be defined more concretely, broadly speaking, the term *adversarial example* refers to an instance that *resembles* one class (e.g., "dog"), while fooling a network into outputting a different class (e.g., "sports car"). While the meaning of "resemblance" is nebulous, we essentially take it to mean that an adversarial example is derived by *perturbing* a properly-classified[1] natural input in a semantically meaningless way—for example, the perturbation may be small enough to be imperceptible to the human eye, or simply inconspicuous in the given context.

Even with this understanding of "resemblance," this definition is somewhat vague. Much of the work addressing adversarial examples has focused on a particular subclass of adversarial example derived from perturbations that can be defined concretely, namely *small-norm adversarial examples* (stated formally in Definition 5.1). Essentially, as the name suggests, small-norm adversarial examples are derived by making a small perturbation (as measured by an ℓ_p norm, e.g., Euclidean distance) to a natural input that causes the network to change its prediction. Geometrically, this means that the original point is near a decision boundary in the model, as shown in Figure 5.1.

Definition 5.1 (Small-norm Adversarial Examples)**.** *Let $F : \mathbb{R}^n \to [m]$ be a neural network classifier, and let $x \in \mathbb{R}^n$ be a natural input to F. We say that an input, $x' = x + \delta$, is an ϵ-small-norm adversarial example with respect to ℓ_p norm $||\cdot||_p$ if*

$$||\delta||_p < \epsilon \quad \wedge \quad F(x) \neq F(x').$$

Besides being possible to easily specify, small-norm adversarial examples are motivated by the fact that for sufficiently small ϵ, the difference between the natural input x and adversarial example x' will be *imperceptible* (in domains for which a notion of perception makes sense, e.g., images, sound waves). Thus, if ϵ is chosen appropriately, we can be assured that the perturbations used to construct such adversarial examples do not alter the original input semantically, as desired by our broader definition. We will provide a deeper discussion

[1]While the literature does not impose a requirement that an adversarial example is derived from a *correctly-classified* point, it *is* assumed that the perturbation causes the network to change its prediction. Moreover, in spirit, to say that a neural network was "fooled" by an adversarial example suggests that its initial behavior on the original point was non-arbitrary.

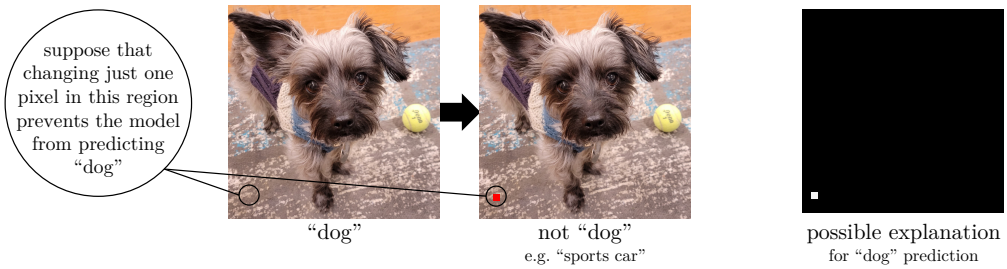


Figure 5.2: Example of how an explanation can be unintelligible, yet justified.

of adversarial examples that fit this criterion despite having large-norm perturbations at the end of Chapter 7.

Clearly, adversarial examples negatively impact the reliability of neural networks that are vulnerable to them—and constitute a security concern in safety-critical machine learning systems—as they lead to unexpected erroneous behavior on seemingly benign inputs. Upon closer inspection, it becomes clear that the existence of adversarial examples in a network must constitute a violation of conceptual soundness as well. Indeed, the perturbation δ, which is semantically meaningless by construction, is nonetheless causally relevant to changing the model's prediction, as witnessed by the anomalous prediction it induces. Thus δ can be thought of as corresponding to some ill-conceived concept that the network erroneously encodes and employs.

Consider the following thought experiment illustrated in Figure 5.2. Suppose we have an image, like the image of a panda on the left in the figure, that our model correctly labels as "dog." Further, let us suppose that changing one pixel in the corner of the image to red leads the model to produce a different label from "dog," e.g., "sports car." If we ask why the model labeled the perturbed image as "sports car," it is clearly reasonable for an explanation to highlight the single red pixel. After all, we have observed that this pixel was causally relevant in changing the model's prediction to "sports car;" and furthermore, there appears to be no other possible reason to label the perturbed image as such.

This accounts for odd explanations on adversarial examples (like the perturbed image that was labeled "sports car"), but what about benign inputs? When we ask the model why it labeled the original image as "dog," we might get an explanation highlighting nothing but the single pixel in the corner of the image that could have been perturbed to produce the adversarial example, as shown in Figure 5.2. While this explanation is certainly confusing, when we consider the context of the model's behavior, there is actually an argument for why such an explanation would be justified. After all, had it not been for the value taken by that pixel in the original image, the label might not have been "dog"—if it had been red, for example, the image would have been labeled as "sports car." For a sufficiently local explanation (e.g., with a single-point distribution of interest), the value of this specific pixel may indeed be the most consequential factor in ensuring the "dog" prediction, as a small change to any other pixel may hardly affect the model's behavior.[2] Thus it is reasonable to consider that pixel to be highly important in the model's decision to label the original image as "dog."

[2]This should not be taken to mean that only local explanations exhibit artifacts relating to adversarial examples. For example, Wang et al. [153] show that Integrated Gradients can be affected by similar anomalies in a model's decision surface.

Relatedly, Ilyas et al. [65] argue that adversarial examples arise because of "non-robust features" that are reliable predictors (i.e., they are correlated with the class labels), *provided that they remain unperturbed*; i.e., they lose their predictive power under small-norm perturbations. As such, these non-robust features are useful for standard classification, but *not* for robust classification. These non-robust features are inherently imperceptible to humans, as they essentially correspond to the perturbations leveraged to create adversarial examples.

This suggests that in order for a model to be interpretable, it must make decisions based on features that, unlike non-robust features, are fundamentally intelligible to humans. Furthermore, we cannot expect that a model will naturally learn human-intelligible features without adequate regularization. After all, there are many ways of using features that could be consistent with the training data, but clearly not all of them are intelligible—indeed, the non-robust features described by Ilyas et al. match this description precisely.

5.2 Provable Robustness Guarantees

Numerous defenses have been proposed to deal with the threat of adversarial examples—particularly small-norm adversarial examples [30, 44, 84, 88, 97, 166]. While many of these approaches are heuristic in nature—that is, they do not provide guarantees that tell us when, or if, the model's predictions cannot be manipulated—in the most safety-critical applications, this may not be good enough, particularly as such solutions have often been subverted by subsequent adaptive attacks [19, 29]. Thus, we will focus on *provable* defenses, which better address the safety concerns raised by adversarial examples, and ensure improved conceptual soundness.

The first question is, *what, precisely, do we want to prove*? To address the case of small-norm adversarial examples, we will target a property called *local robustness* (Definition 5.2). Local robustness on a point, x, stipulates that all points within a distance of ϵ from x are given the same label as x. Geometrically, this means that a ball of radius ϵ around x is guaranteed to be clear of any decision boundaries. From this intuition, it should be clear that local robustness at x precludes the possibility of deriving an adversarial example from x.

Definition 5.2 (Local Robustness)**.** *A model,* $F : \mathbb{R}^n \to [m]$*, is* ϵ-locally-robust *at point,* $x \in \mathbb{R}^n$*, with respect to* ℓ_p *norm,* $||\cdot||$*, if*

$$\forall x' \in \mathbb{R}^n \quad . \quad ||x - x'|| \leqslant \epsilon \implies F(x) = F(x').$$

As the name suggests, local robustness is *local* in the sense that it applies to the neighborhood of a single point, x. We are interested more broadly in *models* that are robust in some way. But this raises the question, *what does it mean for an entire network to be robust*? Perhaps the most straightforward answer to this question would be to propose that a network, F, is robust if F is locally-robust at x for all $x \in \mathbb{R}^n$. However, upon inspection, we see that no interesting network can satisfy such a property: if the network contains a boundary at all, some points must be arbitrarily near it. In other words, a network that is robust in this sense must degenerately make the same prediction everywhere—not a very "interesting" network.

This suggests that we must admit the network to not be locally robust in some parts of the input space; however, this does not have to present a problem as long as the network is robust in the places we care about—the data manifold. For example, an image that bears

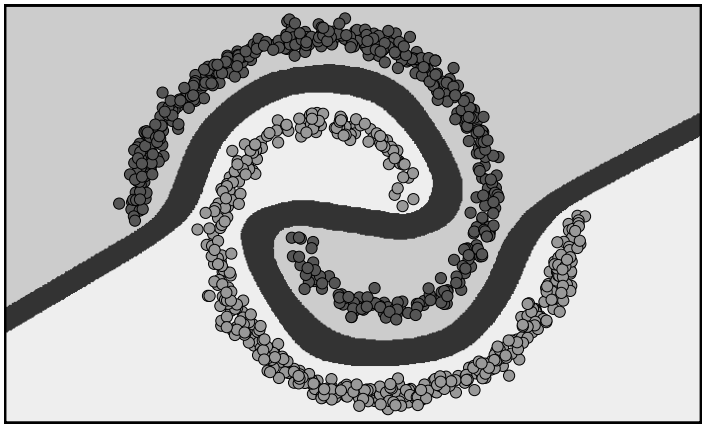


Figure 5.3: Illustration of global robustness. The model abstains from predicting on the margin between the classes (dark gray), which has width at least ϵ.

no resemblance to *any* class (e.g., an image consisting of pure noise) will still be categorized as *some* class by the network; yet there is no particular reason to insist that this arbitrary decision be robust to perturbations that simply produce other ill-formed inputs. Moreover, if we assume that our class labels can be applied unambiguously to every valid (on-manifold) input, then we have reason to believe the "boundary" of the ground truth does not contain nearby valid inputs—we will return to this point shortly.

While a model cannot be locally-robust on its entire input space, it may contain *contiguous robust regions*, R, over which the model is ϵ-locally-robust at every point $x \in R$. Such regions must contain only points of a single class, i.e., $F(x) = F(x')\ \forall x, x' \in R$. Additionally, for any two robust regions, R_1 and R_2, receiving different labels from F, all points in R_1 must be a distance of at leas 2ϵ from points in R_2. This follows from the fact that R_1 and R_2 must be on opposite sides of some decision boundary, and points in each set are a distance of at least ϵ from all boundaries. For a given boundary, if we were to extend all contiguous robust regions maximally, we would obtain a margin of width 2ϵ centered on the boundary. In order to distinguish between points that belong to some robust region and those that do not, we will introduce an additional class, $\perp$, which represents points that are not locally robust. If we assume that the data are 2ϵ-separated, i.e., on-manifold points with different ground-truth labels are a distance at least 2ϵ from one another, then the entire data manifold may be contained within the appropriately labeled robust regions.

We will use this intuition as inspiration to define a more formal notion of *global robustness* (Leino et al. [88]), which extends the concept of local robustness to models. At a high level, we will consider a model to be globally-robust as long as it separates regions of the input space that are given different non-$\perp$ labels by a distance of at least 2ϵ. An illustration of a globally-robust decision boundary is shown in Figure 5.3.

First, let us define the relation, $\stackrel{\perp}{=}$, over classes from $[m] \cup \{\perp\}$, given by Definition 5.3.

Definition 5.3 ($\stackrel{\perp}{=}$)**.** *For $c_1, c_2 \in [m] \cup \{\perp\}$, we say that $c_1 \stackrel{\perp}{=} c_2$ if*

$$c_1 = \perp \quad \vee \quad c_2 = \perp \quad \vee \quad c_1 = c_2.$$

Using this relation, we provide our formal notion of global robustness in Definition 5.4.

Definition 5.4 (Global Robustness)**.** *A model,* $F : \mathbb{R}^n \to [m]$*, is* ϵ-globally-robust*, with respect to* ℓ_p *norm,* $||\cdot||$*, if*

$$\forall x_1, x_2 \in \mathbb{R}^n \quad . \quad ||x_1 - x_2|| \leqslant 2\epsilon \implies F(x_1) \triangleq F(x_2).$$

Global robustness can also be thought of as capturing the operational properties of online local robustness certification. That is, suppose we have some procedure, `cert`, which, given a model, F, and point, x, tells us if F is ϵ-locally-robust at x.[3] Then, we can obtain a globally-robust model, F', from F by adhering to the following procedure: if $\texttt{cert}(F, x, \epsilon)$, then output $F(x)$, otherwise output $\perp$. This procedure essentially adds a $\perp$-labelled region of width at least ϵ to either side of an underlying model's decision boundary, resulting in the requisite 2ϵ. Conversely, given an ϵ-globally-robust model, F', if we let $\mathcal{X} = \{x : F'(x) \neq \perp\}$ there exists some corresponding "underlying" model, F, such that $\forall x \in \mathcal{X}$, $F(x) = F'(x)$ and F is ϵ-locally-robust at x; however the construction of F from F' is less straightforward than producing F' from F.

Finally, we remark that while global robustness can be trivially satisfied by predicting $\perp$ on all points, it is important to keep in mind that the goal of robustness does not replace the overarching goal of accuracy—shared by all neural classifiers. Thus, the utility of a globally-robust model depends on labeling only points off the data manifold as $\perp$, as illustrated in Figure 5.3. However, regardless of whether this goal is entirely met, globally-robust networks warn us when local robustness cannot be guaranteed.

We typically measure the success of a globally-robust model in terms of its *verified robust accuracy* (VRA), which empirically counts the fraction of points from some collection of validation data that are correctly classified by a non-$\perp$ label (i.e., for which the underlying model is both correct and guaranteed to be locally robust). This represents a lower bound on the accuracy the underlying model could achieve under manipulation by a norm-bounded adversary, and serves as a better quality measure of the model's predictions than standard accuracy provides.

Robustness as a Starting Point for Conceptual Soundness. Viewed from the framework of Ilyas et al. described in Section 5.1, small-norm adversarial examples reflect a model's use of non-robust features [65]. While these features are valuable for classification on benign inputs, they have no utility for the objective of robust classification, as they open the door to adversarial examples. Non-robust features inherently pose a problem for conceptual soundness as they are by their very nature uninterpretable and semantically meaningless. If non-robust features are incompatible with conceptual soundness, and are by all accounts ubiquitous in standard deep networks, then robustness—which precludes their use—is an essential property for conceptual soundness. Indeed, any adversarial example is a witness to unsound behavior. Prior work already provides evidence for this claim, finding that explanations—including simple point-gradients—produced on empirically robust models look more intuitive and align more closely with the salient objects in image classification tasks [37, 65, 148, 154]. In Chapter 6, we will discuss how robust models can be obtained.

[3] For this analogy, we need only require that `cert` is *sound*, i.e., it is acceptable for `cert` to reject robust points as long as it does not accept non-robust points.

Chapter 6

Globally Robust Neural Networks

Because of the threat posed by adversarial examples, and the elusiveness of such attacks against heuristic defenses [19, 29], the goal of obtaining local robustness guarantees—to rigorously rule out small-norm adversarial examples—has been sought with intense effort. Within the realm of methods for guaranteeing robustness, we will focus our initial discussion on two classes of approaches. On the one hand, we have methods for post hoc local robustness certification [44, 69, 71, 144, 155, 168, 169], which attempt to derive point-wise guarantees on a pre-trained model. Of course, adversarial examples are known to be ubiquitous in models not specifically trained to resist them [48, 111, 141], meaning that post hoc certification will not find success on standard-trained models. However, many heuristic defenses—primarily based upon *adversarial training* [97], which exposes the model to adversarial examples during training—produce models for which post hoc certification may, in theory, provide meaningful guarantees (provided they have stood up to existing attacks [29]).

Post hoc certification faces several significant challenges. First, we cannot assume *a priori* that any particular pre-trained model achieves a meaningful degree of local robustness in the first place. Indeed, it remains unclear whether the types of heuristic defenses used in concert with post hoc certification truly yield models that will withstand novel attacks. If they do not, any sound certification routine will overwhelmingly reject inputs, destroying the utility of the model. However, even when the underlying model is locally robust at a point under scrutiny, proving this (in ReLU networks) has been shown to be an NP-complete problem [71, 133]. As a result, post hoc certification methods have been forced into a trade-off between precision and efficiency.

Some methods perform *complete* certification, which exactly decides whether a given point is locally robust at radius ϵ [69, 71, 144]. Because of the fundamental difficulty of this problem, these methods cannot scale beyond small "toy" networks. Fromherz, Leino et al. [44] proposed a slight relaxation of complete certification that remains practically precise, but with improved speed and scalability. Still, this approach is only viable for networks of a few thousand neurons, while state-of-the-art networks contain *millions* of neurons. On the opposite end of the spectrum, other work has opted for efficient, but loose, over-approximate certification [155, 168]. While these methods scale to much larger networks, their robustness guarantees are highly-conservative; e.g., Fromherz, Leino et al. found that the certified

radii of some such approaches may underestimate the true robustness by multiple orders of magnitude [44].

In short, post hoc certification is either discouragingly inexact or prohibitively expensive. This has led the research community towards the conclusion that specialized training procedures are likely necessary to obtain formal guarantees on larger state-of-the-art architectures [44]. For example, Croce et al. [30] and Xiao et al. [159] proposed regularization methods for producing models that are more amenable to efficient complete certification. Other methods seek to construct models on which over-approximate certification is more precise [82, 88, 91, 103, 147, 157, 169].

In our work, we similarly direct our focus towards training procedures that produce certifiably robust models. Specifically, we contribute a type of network that we call a *GloRo Net*—short for **Glo**bally **Ro**bust **Net**work—that is globally robust (Definition 5.4) *by construction*. As certification is baked directly into the network, it is natural to optimize GloRo Nets for verified robust accuracy (VRA). In the remainder of this chapter, we cover the construction, implementation, and evaluation GloRo Nets.

6.1 Constructing Globally-Robust Networks

Recall that *global robustness* (Definition 5.4) requires that we impose a thickness of 2ϵ on the boundary of a network, where points landing within this "thick" boundary are labeled as $\bot$, flagging them as not locally robust. To capture this directly, we instrument a model with an extra $\bot$ output that will be applied to points near decision boundaries. In particular, we aim to assign the $\bot$ label such that the instrumented model predicts a non-$\bot$ class *only if the point is ϵ-locally-robust* (with respect to the original model). Essentially, this $\bot$ output is obtained by applying an on-line local robustness check as each input is evaluated; however, the key to our approach is that this check is incorporated naturally into the network's architecture. This means that (1) the network is always globally robust *by construction*, and therefore (2) the joint goal of accuracy and robustness can be reduced to simply training the instrumented model to make the correct predictions (which means not predicting $\bot$).

Before explaining our approach, we will first briefly take an aside to establish a high-level understanding of *why* small-norm adversarial examples occur in certain models, as these insights will be key to our implementation of globally-robust neural networks.

Lipschitz Continuity: the Key to the Avoiding Adversarial Examples

If we are to achieve the ideal for globally-robust models described in Chapter 5—applying the $\bot$ label to only points off the data manifold (e.g., see Figure 5.3)—our model must not contain small-norm adversarial examples derived from on-manifold points, as these will always be labeled as $\bot$ by any globally-robust model.

Recall from Definition 5.1 that for some point x on the data manifold, an ϵ-small-norm adversarial example takes the form $x' = x + \delta$, where $F(x) \neq F(x')$ and $||\delta|| < \epsilon$. For simplicity, we will consider a *binary classifier*, where the network function, f, has only one output and $F(x)$ is determined by the sign of $f(x)$. Supposing that the model's original prediction on x was *confident* (i.e., $|f(x)| \gg 0$)—indeed, in practice, even confident predictions are subject to adversarial manipulation [141]—the difference in the network's output on x and x', $|f(x) - f(x')|$ will be large. Meanwhile, the distance between x and x', $||\delta||$, is small. Thus, the average slope of the function between x and x', given by Equation 6.1,

will be especially large.

$$\frac{|f(x) - f(x')|}{||x - x'||} = \frac{|f(x) - f(x')|}{||\delta||} \tag{6.1}$$

Thus, we see that adversarial examples can be derived from confidently-labeled points only when the network function contains problematically-steep regions. Meanwhile, as we and others have observed, *Lipschitz continuity* (Definition 6.1)—specifically *local* Lipschitz continuity—is a key component of robustness [48, 88, 155, 160]. Essentially, Lipschitz continuity stipulates that the maximum steepness of a function is bounded by some constant, K, known as the *Lipschitz constant*. When the Lipschitz constant is small, we can be assured that the function is not too steep in *any* region.

Definition 6.1 (Lipschitz Continuity)**.** *A function, $h : \mathcal{X} \to \mathbb{R}$, is K-Lipschitz-continuous with respect to ℓ_p norm, $||\cdot||$, if*

$$\forall x_1, x_2 \in \mathcal{X} \quad . \quad \frac{|h(x_1) - h(x_2)|}{||x_1 - x_2||} \leqslant K.$$

A direct corollary of Lipschitz continuity that we will use in our construction is that a change in the function's input of at most ϵ (measured in ℓ_p space) can induce a change in the function's output of *at most* ϵK.

Constructing Globally-Robust Networks

The logit outputs of a neural network, f, can be thought of as m individual functions, f_i, for $i \in [m]$. The prediction, j, of the network on a particular point, x, corresponds to the largest such logit output, i.e., $j = \text{argmax}_i\{f_i(x)\} = F(x)$. Intuitively, if none of the logit functions is arbitrarily steep, then if $f_j(x)$ is sufficiently large compared to any of the other logit outputs, a small perturbation to x will not be able to account for this difference. Specifically, let $\Delta_i(x) = f_j(x) - f_i(x)$, for $i \neq j$, be the size of the margin between the highest logit output and the i^{th} logit output. While $\Delta_i(x)$ is measured in the output space, we can ensure it is sufficiently large to ensure local robustness by relating the output space to the input space via the Lipschitz constant of each logit output.

Namely, let K_i be an upper bound on the Lipschitz constant of f_i. Recall that K_i bounds the largest possible change in the logit output for class i per unit change in the model's input. Thus, on any input x' within a distance of ϵ from x, f_j can *decrease* by at most ϵK_j and any other logit, f_i, can *increase* by at most ϵK_i. Together, these observations suggest that Δ_i can decrease by at most $\epsilon(K_j + K_i)$. If $\Delta_i(x')$ becomes *negative* then some logit besides f_j must have surpassed f_j, meaning that the network must have changed its prediction on x'. On the other hand, if $\Delta_i(x')$ remains positive for all x' in the ϵ-ball around x, then the model's prediction on x is preserved, meaning the network is locally robust at x.

Therefore, let us define our instrumented model, or *GloRo Net*, which we will denote by $\bar{f}^{\epsilon}$, as follows: Let $j = F(x)$, i.e., the class predicted on point x. Let $y_{\perp}(x)$, the logit corresponding to the $\perp$ class, be defined according to Equation 6.2. In essence, $y_{\perp}(x)$ captures the logit value that most competitive class with j would take under the worst-case change to x within an ϵ-ball (accounting for the possible decrease in f_j). Figure 6.1 provides an illustration of this construction.

$$y_{\perp}(x) := \max_{i \neq j} \{f_i(x) + \epsilon(K_i + K_j)\} \tag{6.2}$$

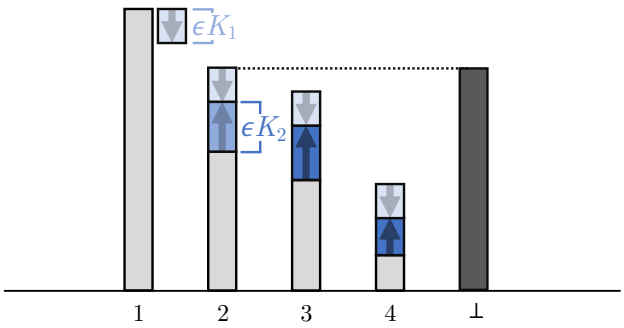


Figure 6.1: Illustration of calculating the $\perp$ logit. Note that ϵK_i provides a bound on changes to logit i within an ϵ-ball. The $\perp$ logit is chosen to account for the predicted class *decreasing* by the maximum amount and each other class *increasing* by the maximum amount. If the $\perp$ logit does not surpass that of the predicted class, then no class can overtake the predicted class within an ϵ-ball (Theorem 6.1).

We then define $\bar{f}^\epsilon$, according to Equation 6.3. Essentially, $\bar{f}^\epsilon$ concatenates $y_\perp$ with the output of f.

$$\forall i \in [m] \cup \{\perp\} \quad . \quad \bar{f}^\epsilon_i(x) := \begin{cases} f_i(x) & \text{if } i \in [m] \\ y_\perp(x) & \text{if } i = \perp \end{cases} \tag{6.3}$$

A key property of a GloRo Net, $\bar{f}^\epsilon$ is that its predictions, $\bar{F}^\epsilon$, can be used to certify the predictions of the instrumented model, F: namely, whenever $\bar{F}^\epsilon$ predicts a class that is not $\perp$, the prediction coincides with the prediction of F, and F is guaranteed to be locally robust at that point (Theorem 6.1).

Theorem 6.1. *If $\bar{F}^\epsilon(x) \neq \perp$, then $\bar{F}^\epsilon(x) = F(x)$ and F is ϵ-locally-robust at x.*

Proof. Let $j = F(x)$. Assume that $\bar{F}^\epsilon(x) \neq \perp$; this happens only if one of the outputs of f is greater than $\bar{f}^\epsilon_\perp(x)$ — from the definition of $f_\perp(x)$, it is clear that only $f_j(x)$ can be greater than $\bar{f}^\epsilon_\perp(x)$. Therefore $f_j(x) > \bar{f}^\epsilon_\perp(x)$, and so $\bar{F}^\epsilon(x) = j = F(x)$.

Now assume x' satisfies $||x - x'|| \leqslant \epsilon$. Let K_i be an upper bound on the Lipschitz constant of f_i. Then, $\forall i$

$$\begin{aligned} & \frac{|f_i(x) - f_i(x')|}{\epsilon} \leqslant \frac{|f_i(x) - f_i(x')|}{||x - x'||} \leqslant K_i \\ \implies \quad & |f_i(x) - f_i(x')| \leqslant K_i \epsilon \end{aligned} \tag{6.4}$$

We proceed to show that for any such x', $F(x')$ is also j. In other words, $\forall i \neq j$, $f_i(x') < f_j(x')$. By applying the definition of the Lipschitz constant as in (6.4), we obtain (6.5). Next, (6.6) follows from the fact that $\bar{f}^\epsilon_\perp(x) = \max_{i \neq j} \{f_i(x) + \epsilon(K_i + K_j)\}$ (Equation 6.2). We then obtain (6.7) from the fact that $f_j(x) > \bar{f}^\epsilon_\perp(x)$, as observed above. Finally, we again

apply (6.4) to obtain (6.8).

$$f_i(x') \leqslant f_i(x) + |f_i(x) - f_i(x')| \leqslant f_i(x) + K_i\epsilon \tag{6.5}$$
$$\leqslant \bar{f}^{\epsilon}_{\perp}(x) - K_j\epsilon \tag{6.6}$$
$$< f_j(x) - K_j\epsilon \tag{6.7}$$
$$\leqslant f_j(x) - |f_j(x) - f_j(x')| \leqslant f_j(x') \tag{6.8}$$

Therefore, $f_i(x') < f_j(x')$, and so $F(x') = j$. We thus conclude that F is ϵ-locally-robust at x. □

Note that in our formulation of GloRo Nets in Equation 6.3, we assume that the predicted class, j, will decrease by the maximum amount within the ϵ-ball, while all other classes increase by their respective maximum amounts. This is a conservative assumption that guarantees local robustness; however, in practice, we can dispose of this assumption by instead calculating the Lipschitz constant of the margin, Δ, by which the logit of the predicted class surpasses the other logits, i.e., the Lipschitz constant of $f_j - f_i$ for $i \neq j$. The details of this tighter variant are presented in Appendix A.1 along with the corresponding correctness proof.

Notice that the GloRo Net, $\bar{F}^{\epsilon}$, will always predict $\perp$ on points that lie directly on the decision boundary of F. Moreover, any point that is within ϵ of the decision boundary will also be labeled as $\perp$ by $\bar{F}^{\epsilon}$. From this, it is perhaps clear that GloRo Nets achieve global robustness (Theorem 6.2).

Theorem 6.2. *$\bar{F}^{\epsilon}$ is ϵ-globally-robust.*

Proof. Assume x_1 and x_2 satisfy $||x_1 - x_2|| \leqslant 2\epsilon$. Let $\bar{F}^{\epsilon}(x_1) = c_1$ and $\bar{F}^{\epsilon}(x_2) = c_2$.

If $c_1 = \perp$ or $c_2 = \perp$, global robustness is trivially satisfied.

Consider the case where $c_1 \neq \perp$, $c_2 \neq \perp$. Let x' be the midpoint between x_1 and x_2, i.e., $x' = (x_1 + x_2)/2$. Thus

$$||x_1 - x'|| = \left|\left|\frac{x_1 - x_2}{2}\right|\right| = \frac{||x_1 - x_2||}{2} \leqslant \epsilon.$$

By Theorem 6.1, this implies $F(x') = c_1$. By the same reasoning, $||x_2 - x'|| \leqslant \epsilon$, implying that $F(x') = c_2$. Thus, $c_1 = c_2$, so global robustness holds. □

6.2 Lipschitz Bounds

The logit corresponding to the $\perp$ class that is added in our construction of GloRo Nets in Section 6.1 is a function of the Lipschitz constant, K_i, of each of the logit outputs of the instrumented model. Thus, far, we have not discussed how this value is obtained. In practice, it is only computationally tractable to obtain an *upper bound* on the Lipschitz constant of the instrumented model. Furthermore, even with the *exact* Lipschitz constant, the $\perp$ logit is a conservative indicator of robustness; that is, although a GloRo Net is guaranteed to be locally robust at any point where it does *not* predict $\perp$—as stated by Theorem 6.1—when the GloRo Net *does* output $\perp$, it is *not* necessarily the case that the point in question is not locally robust. As we will see, this looseness can be mitigated by learning model parameters for which GloRo certification is nonetheless effective. However, this means that the ability

to derive rich gradient information from the Lipschitz bound computation is instrumental to the success of our approach.

Section 6.2.1 provides deeper insight into how the seemingly naive approach of using a simple upper bound on the Lipschitz constant can yield shockingly good results. We then describe the precise details behind how we compute Lipschitz bounds in Section 6.2.2.

6.2.1 Revisiting the Global Lipschitz Constant

The *global* Lipschitz constant (otherwise known simply as the Lipschitz constant) gives a bound on the maximum rate of change in the network's output over its entire input space. For the purpose of certifying robustness, however, it suffices to bound the maximum rate of change in the network's output over any pair of points *within the ϵ-ball* centered at the point being certified, i.e., the *local* Lipschitz constant [160]. In fact, recent work has explored methods for obtaining upper bounds on the local Lipschitz constant [82, 155, 168]; and the construction of GloRo Nets given in Section 6.1 remains correct whether K represents a global or a local Lipschitz constant.

The advantage to using a local bound is, of course, that we may expect tighter robustness guarantees; after all, the local Lipschitz constant is no larger than the global Lipschitz constant. However, using a local bound also has its drawbacks. First, a local bound is typically more expensive to compute. In particular, a local bound always requires more memory, as each instance has its own bound, hence the required memory grows with the batch size. This in turn reduces the amount of parallelism that can be exploited when using a local bound, reducing the model's throughput.

Furthermore, because the local Lipschitz constant is different for every point, it must be computed every time the network sees a new point. By contrast, the global bound can be computed in advance, meaning that verification via the global bound is essentially free. This makes the global bound advantageous, assuming that it can be effectively leveraged for verification. Note however, that this advantage only applies *after training*, when the model parameters have been fixed.

It may seem initially that a local bound would have greater prospects for successful certification. First, *local* Lipschitzness is sufficient for robustly classifying well-separated data [160]; that is, global Lipschitzness is not necessary. Meanwhile, global bounds on typical networks have been found to be prohibitively large [155], while *local* bounds on in-distribution points may tend to be smaller on the same networks. However, the potential disadvantages of a global bound become less clear if the model is specifically trained to have a small global Lipschitz constant.

In particular, GloRo Nets that use a global Lipschitz constant will be penalized for incorrect predictions if the global Lipschitz constant is not sufficiently small to verify its predictions (as they will output $\bot$ in such cases); therefore, the loss function actively discourages any unnecessary steepness in the network function (even off-manifold). In practice, this natural regularization of the global Lipschitz constant may serve to make the steepness of the network function more uniform, such that the global Lipschitz constant will become similar to the local Lipschitz constant.

We show that this is possible in theory, in that for any network for which local robustness can be verified on some set of points using the local Lipschitz constant, there exists a model on which the same points can be certified using the global Lipschitz constant (Theorem 6.3). This suggests that if training is successful, our approach has the same potential using a global bound as it does using a local bound.

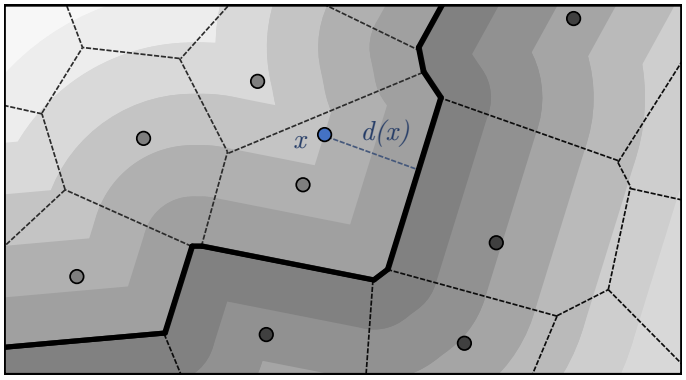


Figure 6.2: Illustration of a function, g, constructed to satisfy Theorem 6.3. The points in S are shown in light and dark gray, with different shades indicating different labels. The Voronoi tessellation is outlined in black, and the faces belonging to the decision boundary are highlighted in bold. The level curves of g are shown in various shades of gray and correspond to points, x, at some fixed distance, $d(x)$, from the decision boundary.

Theorem 6.3. *Let f be a binary classifier that predicts $1 \iff f(x) > 0$. Let $K_L(x, \epsilon)$ be the local Lipschitz constant of f at point x with radius ϵ.*

Suppose that for some finite set of points, S, $\forall x \in S$, $|f(x)| > \epsilon K_L(x, \epsilon)$, i.e., all points in S can be verified via the local Lipschitz constant.

Then there exists a classifier, g, with global Lipschitz constant K_G, such that $\forall x \in S$, (1) g makes the same predictions as f on S, and (2) $|g(x)| > \epsilon K_G$, i.e., all points in S can be verified via the global Lipschitz constant.

Theorem 6.3 is stated for binary classifiers, though the result holds for categorical classifiers as well. Details on the categorical case and the proof of Theorem 6.3 can be found in Appendix A.2; however we provide the intuition behind the construction here. The proof relies on the following lemma, which states that among locally-robust points, points that are classified differently from one another are 2ϵ-separated.

Lemma 6.4. *Suppose that for some classifier, F, and some set of points, S, $\forall x \in S$, F is ϵ-locally-robust at x. Then $\forall x_1, x_2 \in S$ such that $F(x_1) \neq F(x_2)$, $||x_1 - x_2|| > 2\epsilon$.*

Proof. Suppose that for some classifier, F, and some set of points, S, $\forall x \in S$, F is ϵ-locally-robust at x. Assume for the sake of contradiction that $\exists x_1, x_2 \in S$ such that $F(x_1) \neq F(x_2)$ but $||x_1 - x_2|| \leqslant 2\epsilon$. Consider the midpoint between x_1 and x_2, $x' = (x_1 + x_2)/2$. Note that

$$||x' - x_1|| = \frac{||x_1 - x_2||}{2} \leqslant \epsilon$$

Therefore, since F is ϵ-locally-robust at x_1, $F(x') = F(x_1)$. By the same argument, $F(x') = F(x_2)$. But this contradicts that $F(x_1) \neq F(x_2)$. ↯ □

To prove Theorem 6.3, we construct a function, g, whose output on point x increases linearly with x's minimum distance to any face in the Voronoi tessellation of S that separates points in S with different labels. In simpler terms, $g(x)$ is simply the distance from x (sign-adjusted to match the label of x) to a decision boundary that is equidistant from points of

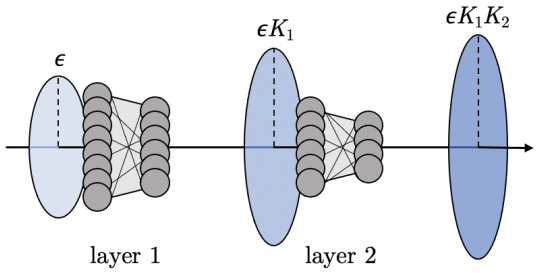


Figure 6.3: Illustration motivating the layer-wise Lipschitz bound (Equation 6.9). In the first stage, we assume that the inputs to the network can be perturbed by a radius of ϵ. If the Lipschitz constant of layer 1 is K_1, this means the corresponding perturbations can be scaled to a radius of at most ϵK_1 after being processed by layer 1. Since the output of layer 1 is the input of layer 2, by the same reasoning, if the Lipschitz constant of layer 2 is K_2, the corresponding perturbations can be scaled to a radius of at most $\epsilon K_1 K_2$ after being processed by layer 2.

either class. An illustration with a general example of g is shown in Figure A.1. Notably, the local Lipschitz constant of g in this construction is everywhere the same as the global constant, making the choice between the two inconsequential.

However, we note that while Theorem 6.3 suggests that networks exist *in principle* on which it is possible to use the global Lipschitz constant to certify 2ϵ-separated data, it may be that such networks are not easily obtainable via training. Furthermore, as raised by Huster et al. [64], an additional potential difficulty in using the global bound for certification is the estimation of the global bound itself. Methods used for determining an upper bound on the global Lipschitz constant, such as the method presented later in Section 6.2.2, may provide a loose upper bound that is insufficient for verification even when the true bound would suffice. Nevertheless, our evaluation in Section 6.4 shows that in practice, the global bound *can* be used effectively for certification (Section 6.4.1), and that the bounds obtained on the models trained with our approach are far tighter than those obtained on standard models (Section 6.4.3).

6.2.2 Computing Lipschitz Bounds

There has been a great deal of work on calculating upper bounds on the Lipschitz constants of neural networks [4, 25, 39, 40, 50, 104, 124, 132, 155, 173]. Our implementation uses the fact that the product of the Lipschitz constant of each of the individual layers of a feed-forward network provides an upper bound on the Lipschitz constant of the entire network [141]. The illustration in Figure 6.3 provides some insight as to why this is the case.

More specifically, the output corresponding to class i of a neural network can be decomposed into a series of k linear transformations, h^i, with $k-1$ interstitial activation functions, σ^i; i.e., $f_i = h^k \circ \sigma^{k-1} \circ h^{k-1} \circ \ldots \circ \sigma^1 \circ h^1$. The Lipschitz constant of an activation function can typically be determined analytically, and in fact, most common activation functions, e.g., ReLU, have a Lipschitz constant of 1. Meanwhile, the Lipschitz constant of each linear layer, h^i, is given by its *operator norm*. In Euclidean space, for linear layers of the form $h_i(x) = xW + b$, this is simply the largest singular value of W, i.e., the largest eigenvalue of $W^T W$. Thus, for 1-Lipschitz-continuous activation functions, Equation 6.9 provides an upper bound on the Lipschitz constant of the i^{th} output of the network f (where $||\cdot||$ is the

operator norm).

$$K_i \leqslant \prod_{j=1}^{k} ||h^j|| \tag{6.9}$$

There are many iterative methods for computing the operator norm of the linear layers, the simplest of which is the power method [39, 50]. Because the power method is *matrix-free*, it allows us to compute operator norm of convolutional layers efficiently, without the need for expanding the kernel to its full matrix form. We will provide more details on practical considerations for calculating the operator norm in Section 6.3.

Gouk et al. also give a procedure for bounding the Lipschitz constant of skip connections, enabling the use of a similar bound to that in Equation 6.9 for ResNet architectures. For more complicated networks, there is a growing body of work on computing layer-wise Lipschitz bounds for various types of layers that are commonly used in neural networks [40, 104, 124, 132, 173].

This calculation must be recomputed as the network parameters change during training; however, after training, the global Lipschitz bound will remain unchanged and therefore it can be computed once in advance. This means that new points can be certified with *no additional non-trivial overhead.*

ℓ_∞ Bounds. While we focus on the ℓ_2 norm in this work, the ideas presented in this Chapter can be applied to other norms, including the ℓ_∞ norm (with appropriate modifications to the Lipschitz bounding procedure [50]). However, we find that the analogue of the approximation of the global Lipschitz bound given by Equation 6.9 appears empirically to be loose in ℓ_∞ space. Meanwhile, a large volume of prior work applies ℓ_∞-specific certification strategies that have proven effective for ℓ_∞ certification [8, 51, 169].

6.3 Training Dynamics

In this section we describe the key implementation details that allow us to obtain state-of-the-art certified accuracy with GloRo Nets.

6.3.1 Loss Functions

Crucially, because certification is intrinsically captured by the GloRo Net's predictions—specifically, the $\bot$ class represents inputs that cannot be verified to be locally robust—a standard learning objective for a GloRo Net corresponds to a robust objective on the original model that was instrumented. That is, we can train a GloRo Net by simply appending a zero to the one-hot encodings of the original data labels (signifying that $\bot$ is never the correct label), and then optimizing a standard classification learning objective, e.g., with cross-entropy loss. Using this approach, $\bar{F}^\epsilon$ will be penalized for incorrectly predicting each point, x, unless x is both predicted correctly *and* F is ϵ-locally-robust at x.

While the above approach is sufficient for training models with competitive VRA, we find that the resulting VRA can be further improved using a loss inspired by TRADES [166], which balances separately the goals of making *correct* predictions, and making *robust* predictions. Recent work [160] has shown that TRADES effectively controls the local Lipschitz continuity of networks. While TRADES is implemented using adversarial perturbations,

which provide an under-approximation of the robust error, GloRo Nets naturally lend themselves to a variant that uses an over-approximation, as shown in Definition 6.2.

Definition 6.2. *(TRADES Loss for GloRo Nets) Given a network, f, cross-entropy loss L_{CE}, and parameter, λ, the GloRo-TRADES loss (L_T) of (x, y) is*

$$L_T(x, y) = L_{CE}\big(f(x), y\big) + \lambda D_{KL}\left(\bar{f}^{\epsilon}(x) || f(x)\right)$$

Intuitively, L_T combines the normal classification loss with the over-approximate robust loss, assuming that the class predicted by the underlying model is correct. Empirically we find that using the KL divergence, D_{KL}, in the second term produces the best results, although in many cases using L_{CE} in both terms works as well.

6.3.2 Gradient Norm Preservation

Despite the fact that Theorem 6.3 states that global-Lipschitz-based certification is sufficiently powerful to match local-Lipschitz-based certification, we may still fail to achieve this result if we cannot accurately calculate the Lipschitz constant of the network. The high-level method presented in Section 6.2.2 (Equation 6.9) provides an upper bound on the Lipschitz constant that is simple to compute, but may be loose. Indeed, Huster et al. [64] prove that no ReLU network can implement the absolute value function in such a way that Equation 6.9 will return a tight bound on the global Lipschitz constant (i.e., 1). More recently, Anil et al. [4] identified *gradient norm preservation* as an important property for obtaining tight layer-wise bounds. A K-Lipschitz layer, $z^i = h^i(z^{i-1})$ of a network, f, is gradient-norm-preserving if it satisfies Equation 6.10; in other words, its Jacobian preserves the norm of incoming gradients during backpropagation, scaling them by its Lipschitz constant.

$$\left\| \frac{\partial f}{\partial z^{i-1}} \right\| = \left\| \frac{\partial f}{\partial z^i} \frac{\partial z^i}{\partial z^{i-1}} \right\| = K \cdot \left\| \frac{\partial f}{\partial z^i} \right\| \tag{6.10}$$

Below, we provide more details on these considerations and their implications regarding our architecture choices for GloRo Nets.

Gradient-Norm-Preserving Activations

ReLU activations—perhaps the most commonly used activation in deep networks—are *not* gradient-norm-preserving. As a result, the bound provided by Equation 6.9 will be loose for ReLU networks that utilize their capacity. Specifically, ReLU networks—and more generally, networks with non-gradient-norm-preserving activations—are faced with an inherent trade-off between non-linear capacity and the tightness of Equation 6.9. Consider the example of the function $g(x) = |x|$. This is a 1-Lipschitz function, and its gradient norm is one almost everywhere. Suppose that Equation 6.9 bounds the Lipschitz constant of some ReLU neural network f as one; i.e., the product of the Lipschitz constants of the linear layers of f is one—this is also a bound on their gradient norm. The Jacobian, of a ReLU activation is a diagonal matrix where each diagonal entry is either one (if the corresponding ReLU switch is active) or zero (if the corresponding ReLU switch is inactive). Thus, the Jacobian tends to shrink the norm of the gradient (which was upper-bounded by one) during backpropagation on x, meaning that the gradient norm of f will be less than one at x. Thus, $f \neq g$. This argument is made more formal in [4, Theorem 1], which states that ReLU networks for

which Equation 6.9 yields a bound of one, and for which the norm of their gradient is one almost everywhere, must be linear.

This result is of particular consequence in light of our construction of the proof of Theorem 6.3, which constructs a function for which the local Lipschitz constant—and in particular the gradient norm—is the same as the global Lipschitz constant everywhere. It suggests that this construction is not achievable with ReLU networks, if we wish to use Equation 6.9 to bound the Lipschitz constant. From this we conclude that gradient-norm-preserving activation functions are necessary for obtaining expressive neural networks that can be tightly certified using the bounds given by Equation 6.9.

Anil et al. propose the *GroupSort* activation, which is gradient-norm-preserving. A special case of the GroupSort activation that is most convenient to use is the *MinMax* activation, which acts on pair of neurons, z_1 and z_2, returning them in sorted order (given by Equation 6.11).

$$\text{MinMax}(z_1, z_2) = \begin{cases} (z_1, z_2) & \text{if } z_1 \leqslant z_2 \\ (z_2, z_1) & \text{otherwise} \end{cases} \tag{6.11}$$

Anil et al. conjecture that any 1-Lipschitz function can be approximated a network using only MinMax activations and linear layers with ℓ_2 operator norms that are bounded by one. We thus use MinMax activations rather than ReLUs for our GloRo Net architectures, finding that this significantly improves the achievable certified performance.

Orthonormal Linear Operators

Anil et al. also argue for orthonormalizing the linear layers of a network. A linear operator is k-orthonormal if its singular values are all equal to k. In dense layers, this corresponds to a weight matrix with orthogonal rows (or columns depending on whether the dimension of the transformation increases or decreases). Such layers essentially perform scaled rotations. Orthonormal operators are gradient-norm-preserving, and may confer additional benefits during training [113]. Thus, several approaches to certifiably robust training have proposed optimizing over the space of orthonormal linear operators [4, 91, 147]—accomplished by constraining the parameters of linear layers to yield only orthonormal transformations. Anil et al. propose a method for orthonormalizing dense layers during training using the Björck algorithm [11]. Orthonormalizing convolutional layers is more difficult, as it does not suffice to orthonormalize the kernels; however, recently Trockman and Kolter [147] proposed a method that acts on the convolution in Fourier space.

However, we find that training GloRo Nets is more successful when the linear layers are *unconstrained*. Specifically, orthonormalized layers appear to inhibit the capacity of the GloRo Net to fit the data, leading to an underfit model. Meanwhile, because the operator norm calculation is incorporated into the GloRo Net's forward pass, gradient information flows back to the linear layer parameters that controls the Lipschitz constant while encouraging accuracy—we discuss this further in Section 6.3.3. As we will see in Section 6.4, GloRo Nets achieve slightly better empirical results than even the most competitive approaches based on orthonormalization, suggesting that a well tailored gradient may lead to sufficient k-othonormality, while making it easier to learn a certifiably robust boundary. We note, however, that the apparent benefits of optimization over orthonormalizing are not yet fully understood, and merit further study.

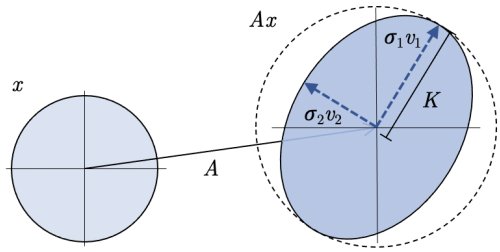


Figure 6.4: Geometric interpretation of the operator norm. Shown in blue on the left is a unit ball in the input space of the operator, A. The image of the unit ball under A is shown in blue on the right, taking the form of an ellipsoid. The eigenvectors of A^*A, v_1 and v_2 give directions for the axes of the ellipse, and the corresponding singular values, σ_1 and σ_2 give the radii. The Lipschitz constant of the operator, K, is given by the largest singular value.

6.3.3 Operator Norm Calculation

Equation 6.9 in Section 6.2.2 gives a way to provide an upper bound on the Lipschitz constant of a neural network based on layer-wise Lipschitz bounds. For linear layers, the layer-wise Lipschitz bound is given by the operator norm of the linear transformation implemented by the layer. Several algorithms exist for computing this operator norm, in selecting the optimal choice, we have three primary considerations: (1) the speed of the computation, (2) the accuracy of the computation, and (3) the gradient signal when backpropagating through the computation. The speed is important to make training practical and efficient; it is not important at test time, however, as once the parameters have been fixed the Lipschitz constant does not need to be recomputed. Conversely, the accuracy is primarily important at test time, while during training it matters only insofar as it affects the gradient of the loss—in a sense, an underestimate of the Lipschitz constant during training is roughly analogous to increasing the temperature of the softmax. Finally, the gradient signal is important for optimization. The proper gradient direction should signal how to decrease the operator norm without shrinking the operator in any unnecessary directions.

To elaborate on this latter point, consider the following geometric interpretation of a linear operator, $A : \mathbb{R}^n \to \mathbb{R}^m$, illustrated in Figure 6.4 for $n = m = 2$. A maps the points in a ball in $\mathbb{R}^n$ to an ellipsoid in $\mathbb{R}^m$. The axes of the ellipsoid are in the direction of the eigenvectors of A^*A, and the principal radii are given by the square root of the corresponding eigenvalues. In order to decrease the operator norm of A, we essentially want to shrink the semi-axis corresponding to the leading eigenvector. If the gradient direction shrinks some linear combination of eigenvectors, the ellipsoid will shrink in multiple directions; this is undesirable, as shrinking in a direction besides that of the leading eigenvector reduces the expressive capacity of the network without reducing the Lipschitz constant. Worse, if the gradient direction causes A to *grow* in the direction of the leading eigenvector, the operator norm will *increase*. Thus we must ensure that the gradient direction captures the direction of the critical eigenvector(s).

With these criteria in mind, we proceed to consider appropriate methods for calculating the operator norm. Although many of the same techniques may apply to both, we will consider the cases of dense and convolutional layers separately.

Algorithm 6.1: The Power Method

Inputs: a linear operator, $A : \mathbb{R}^n \to \mathbb{R}^m$; a number, N of iterations
Output: the operator norm of A

```
def createAttackModel (A , N):
    // Randomly initialize x as a unit vector.
    x ← randomInitialization()
    x ← x/||x||
    for i ∈ [N] do
        x ← A*Ax
        x ← x/||x||
    return ||Ax||
```

Dense Layers

Perhaps the simplest method for computing the operator norm is the power method, given by Algorithm 6.1. One benefit of the power method is that it is matrix-free, meaning that an explicit representation of A is not necessary as long as we have a method of evaluating Av and A^*v' for vectors $v \in \mathbb{R}^n$ and $v' \in \mathbb{R}^m$. The power iterate, x, in Algorithm 6.1 converges to leading eigenvector of A^*A; however, this may take many iterations. Other matrix-free algorithms exist for iteratively computing the leading eigenvector, such as the *Locally Optimal Block Preconditioned Conjugate Gradient* (LOBPCG) algorithm. While the LOBPCG algorithm takes advantage of the fact that A^*A is Hermitian to ensure faster convergence, unfortunately, we were unable realize practical benefits from a naive implementation of LOBPCG, as iterations of LOBPCG are significantly slower than iterations of the power method.

Although the power method is matrix-free, for dense layers, A *is* in fact represented explicitly as a matrix, W. We can thus compute the operator norm by directly computing the eigenvalues of W^TW. Since W^TW is real-symmetric, it is also Hermitian, meaning that we can use specially tailored algorithms that exploit this property. TensorFlow has a native `eigh` function, which returns the eigenvectors and eigenvalues of a Hermitian matrix. While this performs unnecessary work, as we only require the leading eigenvalue, we find this native operation is often quite efficient. Moreover, it is numerically stable, and consistently gives accurate estimates of the eigenvectors and corresponding eigenvalues, as compared to the power method. As the eigenvector is necessary for determining the gradient direction, its accuracy is crucial to successful optimization.

Table 6.5a shows the average runtime for computing forward and backward passes of the power method and TensorFlow's native `eigh` on a 400×600 weight matrix, as well as the accuracy of the resulting eigenvector, x, measured as the cosine similarity between x and W^TWx. We see that the native TensorFlow `eigh` implementation stands out as the fastest method for precisely estimating the leading eigenvalue.

This also translates to `eigh` providing the most reliable gradient signal. Figure 6.6a plots how the operator norm, λ, of W changes as W is modified in the direction of $\frac{\partial \lambda}{\partial W}$, for various methods of estimating λ. We see that λ is reduced most effectively when the gradient is calculated using `eigh`. Although it is not conveyed by Figure 6.6a, we also found that the quality of the power method gradient was far less consistent—sometimes leading to an *increase* in λ—even for large numbers of iterations. Therefore, we use `eigh` rather than

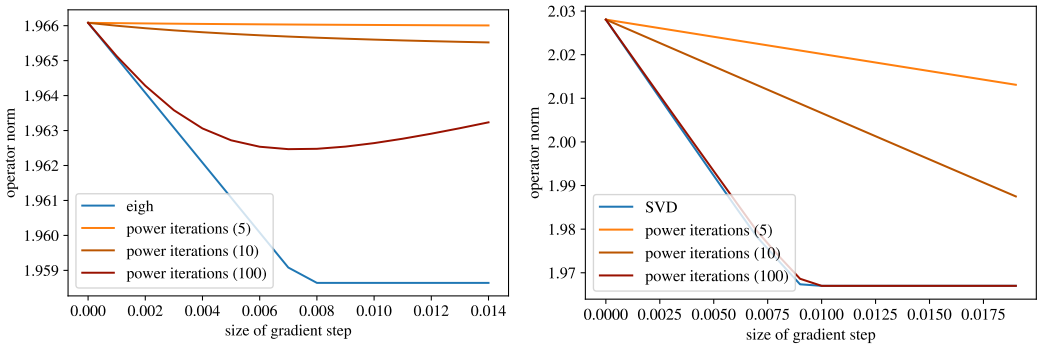
method	*time (s)* *forward pass*	*backward pass*	*accuracy*
power ×5	0.013	0.024	0.993
power ×10	0.024	0.036	0.998
power ×100	0.220	0.300	1.000
`eigh`	0.023	0.030	1.000

(a) dense layer

method	*time (s)* *forward pass*	*backward pass*	*accuracy*
power ×5	0.006	0.009	0.995
power ×10	0.011	0.014	0.998
power ×100	0.096	0.118	1.000
SVD	0.050	0.125	1.000

(b) convolutional layer

Table 6.5: Comparison of the computation speed and eigenvector accuracy of the power method (with various numbers of iterations) and TensorFlow's native `eigh` and SVD implementations on a randomly initialized **(a)** dense layer and **(b)**. Accuracies are measured as the cosine similarity between the eigenvalue estimate, x, and A^*Ax. Note that if x is a true eigenvector of A^*A, then $A^*Ax = \lambda x$.



(a) dense layer

(b) convolutional layer

Figure 6.6: Comparison of the effectiveness of the gradient of various operator norm computations at decreasing the operator norm on a randomly initialized **(a)** dense layer and **(b)**. The x-axis represents the distance the layer weights are perturbed in the direction of the gradient of each of the four computation methods—the power method with 5, 10, and 100 iterations, and TensorFlow's native `eigh` or SVD implementation—and the y-axis measures the corresponding operator norm of the perturbed weights.

the power method for computing the operator norm of dense layers in our implementation.

Convolutional Layers

As mentioned previously, since the power method is matrix-free, it is applicable to convolutional layers in a straightforward way. Specifically, Av can be computed by a convolution of the kernel, W, with v. Analogously A^*v' can be computed via a *transposed convolution*—often referred to as a deconvolution, though this should not be confused with the inverse of the original convolution operation.

As convolutional layers are parameterized by a kernel rather than a full-dimensional weight matrix, `eigh` cannot be directly applied as in dense layers. However, Sedghi et al. [124] demonstrate that the kernel can be mapped into a matrix, M in Fourier space where the singular values of M are the singular values of the convolution operation, A. Experimentally, we found that calling TensorFlow's `eigh` on M^*M was less numerically stable than using

the native TensorFlow SVD implementation, which operates directly on M.

Table 6.5b and Figure 6.6b compare the performance of the power method to that of TensorFlow's SVD implementation on a convolutional layer with 32×32 feature maps and 32 input and output channels. In contrast to what we observed on dense layers, SVD is less comparably favorable. First, Table 6.5b shows that SVD is relatively more costly than the power method. Additionally, Figure 6.6b shows that the gradient signal of the power method is higher quality on convolutional layers than on dense layers—and we also observe that it is much more robust to the random initialization. This difference is likely because the eigenvectors of convolutional layers are typically low-dimensional compared to their counterparts in dense layers—the former have their dimension determined by the kernel, which is typically small.

Another advantage to the power method is that if, rather than initializing x randomly, we begin with a close approximation of the true leading eigenvector, then the power method converges much more rapidly. Thus, we can preserve the state of the power iterate over the course of training to accelerate convergence. While the eigenvector may drift as we update the weights, we can over-provision the number of power iterations and add an early-stopping criterion to terminate iteration once the eigenvector estimate is sufficiently accurate. We base our stopping criterion on the cosine similarity between the values taken by x at the start and end of each iteration—we note that this corresponds to the measurement of cosine similarity between x and A^*Ax used in Table 6.5b. The results from Table 6.5b show that the equivalent of 100 power iterations has been reached once the cosine similarity is within approximately four decimal places from 1.0. This does not affect the gradient computation, as the gradient depends only on the final value of x regardless of the number of iterations—the gradient of the power method is the same as the gradient of $||Ax||$, where x is treated as a constant.

Given these advantages, and the results in Table 6.5b and Figure 6.6b, we opt to use the power method for computing the operator norm of convolutional layers.

6.4 State-of-the-Art Robust Classification

In this section, we present an empirical evaluation of our proposed GloRo Nets. We first compare GloRo Nets with several certified training methods from the recent literature in Section 6.4.1. We also explore the training cost, in terms of per-epoch time and peak memory usage, required for training and certification using our method, compared with other styles approaches (Section 6.4.2). We end by demonstrating that the simple method we use for estimating Lipschitz bounds is surprisingly tight for GloRo Nets (Section 6.4.3).

Experimental Setup

We begin by providing an overview of the methods we compare against and the setup of our experiments.

Certifiable Training Methods. We compare the performance of GloRo Nets to a wide array of certifiable training methods that follow a few high-level approaches. A detailed discussion of these approaches and their relation to GloRo Nets is provided in Section 6.5. First, we consider other methods that perform certification via global Lipschitz bounds,

including Cayley [147], BCOP [91], and LMT [149]. We also consider BCP [82], which uses a *local* Lipschitz bound to perform certification. Finally, we consider two methods performing non-Lipschitz-based certification, KW [157] and MMR [30].

Datasets. Our evaluation considers three benchmark datasets, MNIST [79], CIFAR-10 [76] and Tiny-Imagenet [78], which have commonly been used to evaluate certified training methods. To facilitate comparison with prior work, we use an ℓ_2 adversarial perturbation budget of either $\epsilon = 0.3$ or $\epsilon = 1.58$ for MNIST, and $\epsilon = 0.141$ for CIFAR-10 and Tiny-Imagenet.

Metrics. For each model considered in our evaluation, we report the *clean accuracy*, i.e., the accuracy without verification on non-adversarial inputs, the *PGD accuracy*, i.e., the accuracy under adversarial perturbations found via the Projected Gradient Descent (PGD) attack [97], and the *verified-robust accuracy* (VRA), i.e., the fraction of points that are both correctly classified *and* certified as robust. Whereas the VRA gives a lower bound on the number of correctly-classified points that are locally robust, the PGD accuracy serves as an upper bound on the same quantity. For the methods besides our own, we report the corresponding best VRAs from the original respective papers when possible, but measure training and certification costs on our own hardware for an equal comparison. However, when training BCP models for MNIST with $\epsilon = 0.3$, we found a different set of hyperparameters that outperforms those given by Lee et al. [82], which we report instead.

Models. We used four standard convolutional network architectures that have been used in most of the prior literature on certifiable training to provide the most accurate comparison to the other methods in our evaluation. As discussed in Section 6.3, we found that MinMax activations [4] performed better than ReLU activations for GloRo Nets; however, some of the methods we compare to used ReLU activations, either because their certification analysis is tailored to ReLUs (BCP, KW, and MMR) or because they were published before MinMax became popularized (LMT). We include additional results in Appendix A.3 that show that GloRo Nets trained with ReLU activations still outperform all of these methods. Further details on the network architectures and experimental hyperparameters are provided in Appendix A.4. An implementation of our approach that can be used to reproduce our results is available on GitHub.[1]

6.4.1 Verified Accuracy

We first compare the VRA obtained by GloRo Nets to the VRA achieved by several other certified training approaches. The results are shown in Table 6.7. As additional reference points, for each dataset we also provide numbers obtained via standard training and Randomized Smoothing (RS) [26] on the same architectures (using ReLU activations). We note that RS provides only a *probabilistic* robustness guarantee, while the other VRAs in Table 6.7 correspond to deterministic guarantees. We further note that the numbers reported for RS appear lower than what is often reported for RS and similar methods in the literature because we use the same small CNN architecture used by the other approaches in our evaluation rather than a large ResNet architecture.

We find that GloRo Nets consistently outperform the previous state-of-the-art deterministic VRA. On MNIST, GloRo Nets outperform all previous approaches with both radii

[1] Code available at `https://github.com/klasleino/gloro`.

method	*clean (%)*	*PGD (%)*	*VRA(%)*
MNIST ($\epsilon = 0.3$)			
standard†	99.2	96.9	0.0
GloRo	**99.0**	**97.8**	**95.7**
BCP	93.4	89.5	84.7
KW	98.9	**97.8**	94.0
MMR	98.2	96.2	93.6
RS*	98.8	-	97.4
MNIST ($\epsilon = 1.58$)			
standard†	99.0	45.4	0.0
GloRo	**97.0**	**81.9**	**62.8**
LMT	86.5	53.6	40.6
BCOP	98.8	-	56.7
BCP	92.4	65.8	47.9
KW	88.1	67.9	44.5
RS*	99.0	-	59.1
CIFAR-10 ($\epsilon = 36/255$)			
standard†	85.7	31.9	0.0
GloRo	**77.0**	**69.2**	**60.0**
LMT	63.1	58.3	38.1
BCOP	72.4	64.4	58.7
Cayley	75.3	67.6	59.1
BCP	65.7	60.8	51.3
KW	60.1	56.2	50.9
RS*	74.1	-	64.2
Tiny ImageNet ($\epsilon = 36/255$)			
standard†	35.9	19.4	0.0
Gloro	**35.5**	**32.3**	**22.4**
BCP	28.8	26.6	20.1
RS*	23.4	-	16.9

Table 6.7: Performance comparisons for a wide set of certifiable training approaches approaches, showing the VRA, PGD accuracy, and clean accuracy of each approach. Best results are highlighted in bold. Standard training is included for the sake of comparison but is marked by a † superscript to denote that it does not lead to certified predictions. Randomized Smoothing (RS) is marked with a * superscript to indicate that it provides only a *stochastic* robustness guarantee rather than a deterministic one.

commonly used for the ℓ_2 norm in prior work ($\epsilon = 0.3$ and $\epsilon = 1.58$). When $\epsilon = 0.3$, the VRA begins to approach the clean accuracy of the standard-trained model; for this bound, GloRo Nets outperform the previous best VRA (achieved by KW) by nearly two percentage points, accounting for roughly 33% of the gap between the VRA of KW and the clean accuracy of the standard model. For $\epsilon = 1.58$, GloRo Nets improve upon the previous best VRA by several percentage points—in fact, the VRA achieved by GloRo Nets in this setting even slightly exceeds that of Randomized Smoothing, despite the fact that RS provides only a stochastic guarantee. On CIFAR-10, GloRo Nets slightly exceed the VRA of the other deterministic approaches, though Cayley is competitive. Finally, on Tiny-Imagenet, GloRo Nets outperform BCP by approximately 2 percentage points, improving the state-of-the-art VRA by roughly 10%. Some other methods, such as KW, have been found to be unable to scale to Tiny-Imagenet due to memory pressure, and overall, results on Tiny-Imagenet are less common in the literature.

method	*sec./epoch*	*# epochs*	*mem. (MB)*
MNIST ($\epsilon = 0.3$)			
standard†	0.3	100	0.6
GloRo	0.9	50	0.7
KW	66.9	100	20.2
BCP	44.8	300	12.6
MNIST ($\epsilon = 1.58$)			
standard†	0.9	42	2.2
GloRo	3.7	300	2.7
KW	138.1	60	84.0
BCP	43.4	60	12.6
CIFAR-10 ($\epsilon = 36/255$)			
standard†	1.8	115	2.5
GloRo	6.9	400	3.6
KW	516.8	60	100.9
BCP	47.5	200	12.7
Tiny-Imagenet ($\epsilon = 36/255$)			
standard†	10.7	58	6.7
GloRo	40.3	800	10.4
BCP	798.8	102	715.2

(a)

method	*time (sec.)*	*mem. (MB)*
CIFAR-10 ($\epsilon = 36/255$)		
GloRo	0.4	1.8
KW	2,515.6	1,437.5
BCP	5.8	19.1
RS*	36,845.5	19.8

(b)

Table 6.8: (a) Training cost in terms of runtime and memory usage. Standard training is included for the sake of comparison but is marked by † superscript to denote that it does not lead to certified predictions. **(b)** Certification timing and memory usage results on CIFAR-10. Randomized Smoothing (RS) is marked with a * superscript to indicate that it provides only a *stochastic* robustness guarantee rather than a deterministic one.

Table 6.7 also shows that GloRo Nets perform favorably in terms of the other metrics we consider. Compared to standard training, all certifiably-robust training approaches typically impose some cost on clear accuracy. While the traded benefits of certifiable training are clear (as standard training provides no guarantees of robustness), it is preferable to have a model that performs well even when it can't be certified. To this end, we find that GloRo Nets also consistently achieve better clean and PGD accuracy than other approaches. The former suggests that GloRo Nets do not achieve better VRA than other approaches by simply trading expressiveness for certifiability. The latter indicates that the higher VRA of GloRo Nets translates to practical benefits against real adversaries, rather than merely reflecting tighter certification on similarly robust points.

6.4.2 Training and Certification Cost

A key advantage to GloRo Nets over most prior approaches is their ability to achieve state-of-the-art VRA using methods that are computationally inexpensive both at train and test time. Most importantly, as discussed in Section 6.2, using a global Lipschitz bound for certification confers performance benefits over using a local bound (e.g., BCP), or other expensive approaches (e.g., KW). Furthermore, as discussed in Section 6.3.3, using the power method in favor of weight orthonormalization (used by BCOP and Cayley) for enforcing global Lipschitzness accrues additional performance gains during training. Our evaluation focuses

particularly on the benefits of global-Lipschitz-based certification, which is responsible for the most significant performance benefit of GloRo Nets—namely, instant certification at test time. To this end, we compare against BCP, which uses local-Lipschitz-based certification, and KW, the better of the two non-Lipschitz-based certification procedures we consider in our evaluation.

Table 6.8a shows the cost of GloRo, KW, and BCP—both in time per epoch and in memory—during training. For comparison, Table 6.8a also provides the same measurements for standard training. All timings were taken on a machine using a Geforce RTX 3080 accelerator, 64 GB memory, and Intel i9 10850K CPU, with the exception of those for the KW [157] method, which were taken on a Titan RTX card for toolkit compatibility reasons. Appendix A.5 provides further details on how memory usage was measured. Because different batch sizes were used to train and evaluate each model, we control for this by reporting the memory used *per instance* in each batch.

We see that KW is the most expensive approach to train, requiring tens to hundreds of seconds per epoch and roughly 35× more memory per batch instance than standard training. BCP is less expensive than KW, but still takes nearly one minute per epoch on MNIST and CIFAR and 15 minutes on Tiny-Imagenet, and uses anywhere between 5-106× more memory than standard training.

Meanwhile, the cost of GloRo Nets is more comparable to that of standard training than of KW or BCP, taking only a few seconds per epoch, and at most 50% more memory than standard training. Because of its memory scalability, we were able to use a larger batch size with GloRo Nets. As a result, more epochs were occasionally required for GloRo training; however, this did not outweigh the significant reduction in time per epoch, as the total time for training was still only at most half of the total time for BCP.

Table 6.8b shows the test-time cost of the same set of approaches both in the time required to certify the entire test set and in the memory used to do so (results given for CIFAR-10). KW is the most expensive deterministic approach in terms of time and memory, followed by BCP. Here again, GloRo Nets are far superior in terms of cost, making certified predictions over 14× faster than BCP with less than a tenth of the memory, and over 6,000× faster than KW. These results show that the GloRo approach is promising not only because of its first-class VRA, but also because of its scalability.

Figure 6.8b also provides the evaluation cost for Randomized Smoothing (RS). As reported in the literature (and also somewhat evident from Figure 6.7), RS and its derivatives typically outperform the VRA achievable by modern deterministic approaches—though the stochastic guarantees they provide are not perfectly comparable. However, we see in Figure 6.8b that GloRo Nets are *several orders of magnitude* faster at certification than RS. GloRo Nets perform certification in a single forward pass, enabling certification of the entire CIFAR-10 test set in *under half a second*; on the other hand, RS requires tens of thousands of samples to provide a confident guarantee on a single instance, reducing throughput by orders of magnitude and requiring over *ten hours* to certify the same set of instances. Thus, unlike GloRo Nets, RS is not suitable for applications that require a "real-time" defense.

6.4.3 Lipschitz Tightness

Theorem 6.3 demonstrates that a global Lipschitz bound is theoretically sufficient for certifying 2ϵ-separated data. However, as discussed in Section 6.2, there may be several practical limitations making it difficult to realize a network that reaches this potential; we now assess how these limitations are borne out in practice by examining the Lipschitz bounds that

method	global UB	global LB	local LB
MNIST ($\epsilon = 1.58$)			
standard	$5.4 \cdot 10^4$	$1.4 \cdot 10^2$	17.1
GloRo	2.3	1.9	0.8
CIFAR-10 ($\epsilon = 36/255$)			
standard	$1.2 \cdot 10^7$	$1.1 \cdot 10^3$	96.2
GloRo	15.8	11.0	3.7
Tiny-Imagenet ($\epsilon = 36/255$)			
standard	$2.2 \cdot 10^7$	$3.6 \cdot 10^2$	40.7
GloRo	12.5	5.9	0.8

Table 6.9: Upper (UB) and lower (LB) bounds on the global and local Lipschitz constant. Lower bounds on the local Lipschitz constant reflect an average over the points in the test set.

GloRo Nets use for certification.

Weng et al. [155] report that an upper bound on the global Lipschitz constant is not capable of certifying robustness for a non-trivial radius. While this is true of models produced via *standard training*, GloRo Nets impose a strong implicit regularization on the global Lipschitz constant. Indeed, Table 6.9—which provides the layer-wise Lipschitz upper bound (from Equation 6.9) on both GloRo Nets and their standard-trained counterparts—shows that the global upper bound is several orders of magnitude smaller on GloRo Nets than on standard networks.

Another potential limitation of using an upper bound of the global Lipschitz constant is the tightness of the bound itself [64]. To investigate this possibility, we compute a lower bound on the network's Lipschitz constant via optimization. Specifically, we solve the optimization problem given by Equation 6.12, where $j_1 = F(x_1)$. The fact that Equation 6.12 considers $f_{j_1} - f_i$ reflects the fact that our GloRo Net implementation measures the Lipschitz constant of the margin between the predicted class and all other classes (see Appendix A.1 for more details).

$$\max_{x_1, x_2} \max_i \left\{ \frac{|f_{j_1}(x_1) - f_i(x_1) - (f_{j_1}(x_2) - f_i(x_2))|}{||x_1 - x_2||} \right\} \tag{6.12}$$

Table 6.9 shows that a lower bound of the Global Lipschitz constant, obtained via optimizing Equation 6.12, reaches an impressive 83% of the upper bound on MNIST, meaning that the upper bound is fairly tight. On CIFAR-10 and Tiny-Imagenet the lower bound reaches approximately 70% and 47% of the upper bound, respectively. However, on a standard model, the lower bound is potentially orders of magnitude looser. These results show there is still room for improvement; for example, using the lower bound in place of the upper bound would lead to roughly a 10% increase in VRA on CIFAR-10, from 58% to 64%. However, the fact that the bound is tighter for GloRo Nets suggests the objective imposed by the GloRo Net helps by incentivizing parameters for which the upper bound estimate is sufficiently tight for verification. Moreover, the decreasing tightness from MNIST to CIFAR-10 to Tiny-Imagenet may also partially reflect looseness in the lower bound as the number of input dimensions increases, causing progressively more difficulty in optimizing Equation 6.12.

Finally, we compare the global upper bound to an empirical lower bound of the local Lipschitz constant. The local lower bound given in Table 6.9 reports the *mean* local Lipschitz

constant found via optimizing Equation 6.12 subject to $||x_1 - x_0|| \leqslant \epsilon$ and $||x_2 - x_0|| \leqslant \epsilon$ for each x_0 in the test set. In the construction given for the proof of Theorem 6.3, the local Lipschitz constant is the same as the global bound at all points. While the results in Table 6.9 show that this may not be entirely achieved in practice, the ratio of the local lower bound to the global upper bound is essentially zero in the standard models, compared to 6-35% in the GloRo Nets, establishing that the upper bound is again much tighter for GloRo Nets. Still, this suggests that a reasonably tight estimate of the local bound may yet help improve the VRA of a GloRo Net at runtime, although this is a challenge in its own right. Intriguingly, GloRo Nets outperform BCP, which optimizes over a *local* Lipschitz bound for certification during training, suggesting that GloRo Nets provide a better objective for certifiable robustness despite using a looser bound during training.

6.5 Related Work

The earliest approaches attempting to address the threat of adversarial examples were essentially heuristics motivated by the contemporary attacks of the time []. Such approaches are routinely broken by adaptive attacks []. Goodfellow et al. [48] and Madry et al. [97] proposed minimizing *adversarial loss*—i.e., taking the worst-case adversarial perturbation into account when determining the model's loss during training—as general method of defense. While the concept of adversarial loss is sound in principle, its implementation has traditionally relied on methods that produce adversarial examples—an under-approximation—and thus so-called *adversarial training* has the tendency to "overfit" to specific attacks, while leaving the possibility for new types of attack to succeed. GloRo Nets can be understood as minimizing adversarial loss via an *over-approximation*, however, this has typically been considered distinct from adversarial training.

A body of work has emerged to seek an end to the cat-and-mouse game of adversarial training through methods that provide provable guarantees against adversarial examples—specifically small-norm adversarial examples. Many methods have been proposed to certify the robustness of networks post hoc [], but as discussed in the introduction of this chapter, such methods quickly meet scalability constraints when the networks are not regularized for the explicit goal of certification. Thus focus has been shifted to training certifiably robust models.

Lipschitz-based Certification

Utilizing the Lipschitz constant to certify robustness has been studied in several instances of prior work. On discovering the existence of adversarial examples, Szegedy et al. [141] analyzed the sensitivity of neural networks using a global Lipschitz bound, explaining models' "blind spots" partially in terms of large bounds and suggesting Lipschitz regularization as a potential remedy.

Lipschitz constants have been applied previously for fast post hoc certification [58, 155, 156]. While our work relies on similar techniques, our exclusive use of the global bound means that no additional work is needed at inference time. However, as our results in Table 6.9 in Section 6.4.3 show, layer-wise Lipschitz bounds may be incredibly loose on typical neural networks; this was corroborated by Fromherz, Leino, et al. [44], who find that the radii certified by post hoc Lipschitz methods are indeed quite loose. Thus, a method for controlling the Lipschitz constant during training is a crucial piece of Lipschitz-based certification.

Gradient-based Lipschitz Control. The closest work in spirit to ours is Lipschitz Margin Training (LMT) [149], which also uses incorporates global Lipschitz bounds into a network's loss to train models that are more certifiably robust. The approach works by adding $\sqrt{2}\epsilon K$ to all logits other than that corresponding to the ground-truth class during training. Despite the fact that the construction of GloRo Nets is only subtly different from that of LMT (as GloRo Nets were developed independently), the differences are consequential. In practical terms, our evaluation shows LMT was outperformed by essentially all of the methods considered in our evaluation, while GloRo Nets achieve the best performance. This stark contrast may be due to several factors, including (1) the construction of GloRo Nets does not excessively penalize logits corresponding to boundaries distant from the point being certified, and (2) GloRo Nets admit a straightforward implementation of TRADES loss (Section 6.3), which we find improves performance.

Lipschitz Control via Orthonormalization Cisse et al. [24] introduced "Perseval networks," which enforce non-increasing Lipschitz bounds on all layers by orthonormalizing their weights. This general approach has been refined more recently [91, 147], particularly after Anil et al. [4] proposed replacing ReLU activations with MinMax activations, allowing networks with Lipschitz-bounded layers form a class of *universal Lipschitz approximators*, that that can approximate any Lipschitz-bounded function over a given domain [4, 25]. The most recent development in orthonormalization-based approaches has been a method proposed by Trockman and Kolter [147] that orthonormalizes convolutional layers applying the Cayley transform to a representation of the convolutional kernel in the Fourier domain. This approach is the most competitive with ours, though our analysis in Section 6.3 suggests that layer-wise Lipschitz control via gradients is a viable option when the computation is sufficiently accurate, and thus our slightly superior results may indicate that there are benefits to the gradient-based approach. Nonetheless, we contend that orthonormalized layers are complimentary to our work, as they can be incorporated into GloRo Nets, a direction which merits further study.

Local Lipschitz Control. Recently, Lee et al. [82] explored the possibility of training networks against local Lipschitz bounds, motivated by the fact that the global bound may vastly exceed a typical local bound on some networks. Theorem 6.3 shows that in principle, the choice between local and global bounds may not matter for robust classification. Moreover, while it is true that the layer-wise bounds computed by Equation 6.9 may be loose on some models, our experimental results suggest that it is possible in many cases to mitigate this limitation by training against such bounds with the appropriate loss. The advantages of doing so are apparent in the cost of both training and certification, where the additional overhead involved with computing tighter local bounds is an impediment to scalability.

Other Certification Approaches

Several other methods have been proposed for training ℓ_2-certifiable networks that are not based on Lipschitz constants. For example, Wong et al. [157] use an LP-based approach that can be optimized relatively efficiently using a *dual network*, Croce et al. [30] and Madry et al. [97] propose training routines based on maximizing the size of the linear regions within a network, and Mirman et al. [103] propose a method based on abstract interpretation.

Randomized Smoothing

The certification methods discussed thus far provide *deterministic* robustness guarantees. By contrast, another recent approach, Randomized Smoothing [26, 80], provides *stochastic* guarantees—that is, points are certified as *robust with high probability* (meaning in this case that the probability can be bounded from below). Randomized Smoothing has been found to achieve better VRA performance than any deterministic certification method, including GloRo Nets. However, GloRo Nets compare favorably to Randomized Smoothing in a few key ways.

First, the fact that GloRo Nets provide a deterministic guarantee is an advantage in and of itself. In safety-critical applications, it may not be considered acceptable for a small fraction of adversarial examples to go undetected; meanwhile, Randomized Smoothing is typically evaluated with a false positive rate around 0.1% [26], meaning that instances of incorrectly-certified points are to be expected in validation sets with thousands of points.

Furthermore, as demonstrated in Section 6.4.2, GloRo Nets have far superior run-time cost. Because Randomized Smoothing does not explicitly represent the function behind its robust predictions, points must be evaluated and certified using as many as 100,000 samples [26], reducing throughput by several orders of magnitude. Meanwhile, GloRo Nets can certify a batch of points in *a single forward pass*.

6.6 Future Directions

The work presented in this chapter offers promising results that we expect can further advance the state-of-the-art with continued innovation. The strengths of GloRo Nets are many. First, we find that GloRo Nets outperform other existing methods in terms of the VRA they achieve with deterministic guarantees. Beyond this, they offer efficient training with the ability to achieve results without the need to calculate local bounds, orthonormalize their weights, sample thousands of points, etc. Finally, unlike several methods, they perform certification at no extra cost at test time, making them highly practical for deployment in systems requiring real-time performance. We see many potentially fruitful future directions for continued effort to improve GloRo Nets. In this section we outline a few of the possibilities.

ResNets. Our evaluation considers relatively small CNN architectures that have become standard in most work on deterministic robustness certification. Nondeterministic methods, i.e., those based on Randomized Smoothing, typically achieve best results on large ResNet architectures—on comparably-small convolutional networks like the ones presented in our evaluation, GloRo Nets are actually competitive with simple Randomized Smoothing [26]. The ability for ResNet architectures to achieve superior clean performance suggests they provide potential for higher VRA as well—provided they can be certified. ResNets present a challenge for deterministic certification because of their size (in terms of parameters) as well as their depth. As we saw in Section 6.4, many certification methods are already computationally expensive and memory-intensive on the smaller models from our evaluation, making it unlikely they would scale to ResNet-sized models. GloRo Nets do not face the same limitations, however, particularly deep models make layer-wise Lipschitz approximation potentially looser. ResNet training can also be unstable, particularly without batch normalization, which we find is not compatible with GloRo training—though there is a

growing body of literature on training "normalizer-free" ResNets with no batch normalization [14, 15, 167]. While we have borrowed some of these aforementioned techniques, we conjecture that the Lipschitzness of GloRo Nets lessens the need for batch normalization in the first place. Instead, we find that stability of GloRo ResNet training is best improved by increasing the number of power iterations to get an accurate gradient signal. To date, we have been able to train GloRo ResNets that match our best performance on CIFAR-10 (with 60.1% VRA), but we believe continued effort will be necessary to unlock the full power of ResNet models.

Detailed Analysis of Training Dynamics. The discussion and analysis in Section 6.3 provides useful insights that we leveraged to improve the performance of GloRo Nets. We believe further analysis along these lines may lead to additional insights. For example, we may uncover better ways of tailoring of the loss function and gradient direction to achieve the goals of GloRo training. Additionally, as noted in Section 6.3.2, we believe it will be useful to better understand the costs and benefits of orthonormalizing the linear layers compared to controlling the Lipschitz constant through gradient updates—the latter, used by GloRo Nets, achieves the best empirical results to date. As a starting point, it will be helpful to determine whether (or to what extent) GloRo training naturally converges towards orthonormalized weights, and what the consequences are if it does not.

Chapter 7

Limitations of Local Robustness

The methods presented in Chapter 6 provide concrete steps towards training models free from one of the most fundamental conceptual soundness pitfalls—adversarial examples. While continued effort will be required in this direction to close the gap between the performance of non-robust models (on unperturbed data) and that of models that can provide provable local robustness guarantees, this problem is likely within the grasp of such efforts. However, the progress of robust models—and moreover, their utility for combating adversarial examples—will fundamentally remain inhibited by the specific guarantees they provide. In particular, we allude to the fact that we have, to this point, focused on guarantees of *local robustness*; in this chapter, we discuss the limitations of such guarantees, and their implications for the greater objective of conceptual soundness.

In broad terms, there are two primary ways in which local robustness may be inadequate. First, and perhaps the more subtle point, is that local robustness may be too strict to be appropriate in certain contexts—even granting that bounding the ℓ_p norm of the adversary's perturbation budget by ϵ is sufficient to render the perturbation *imperceptible*. We continue this point in considerable depth in Section 7.1, and propose and evaluate two possible relaxations to address the issue in Sections 7.2-4.

Second, the more glaring and fundamental shortcoming of local robustness is that it addresses only *small-norm* adversarial examples (Definition 5.1). While these are an important class of adversarial examples that minimally capture *imperceptibility*, we must bear in mind that, as mentioned in Chapter 5, adversarial examples may be derived by perturbations that are quite perceptible, yet otherwise semantically irrelevant or inconspicuous. Moreover, though local robustness—and robustness to adversarial examples in general—is an essential cornerstone for conceptual soundness, robustness does not, on its own, imply conceptual soundness. We conclude this chapter in Section 7.5 with a further discussion of these concerns, highlighting the many avenues for future work on robustness, and foreshadowing the handful of conceptual soundness violations orthogonal to robustness that will be addressed in Part III.

7.1 Relaxations of Local Robustness

The objective of robust classification is typically captured by ensuring that a model satisfies point-wise *local robustness*; i.e., given a point, x, the model's predictions must remain

invariant over the ϵ-ball centered at x (Definition 5.2). As discussed in Chapter 5, this can be extended to a notion of *global robustness*, which applies to the underlying model. Robust models are evaluated by a metric, verified-robust accuracy (VRA), which corresponds to the fraction of points that are both correctly classified and locally robust.

In some contexts, however, it is not always clear that VRA is the most desirable objective or the most natural metric for measuring a model's success against adversaries. For example, in some contexts, *not all adversarial examples are equally bad*—this may reflect simply that a mistake is understandable, even if it was caused by an adversary (e.g., if a model mistakenly predicts an image of a leopard to be a jaguar); or that the correct label may be arbitrary in certain cases (e.g., if an image contains two classes of subjects[1]). If only *non-arbitrary* "mistakes" can be induced by an adversary, such changes in prediction may simply be driven by small-magnitude, sound features that merely tip the balance between plausible labels. In such cases, restricting the ability of the model to signal reasonable uncertainty may be counterproductive to achieving conceptual soundness.

For similar sorts of reasons, some computer vision tasks often use *top-k accuracy* as a benchmark metric that relaxes standard accuracy, allowing a prediction to be considered correct so long as the correct label appears among the model's k highest logit outputs. In many applications, top-k accuracy is considered a more suitable metric/objective than standard top-1 accuracy [10]; meanwhile, studied robustness properties are typically only defined with respect to a *single* predicted class. Thus, as part of this work, we introduce an analogous relaxation of local robustness to top-k accuracy, which we call *relaxed top-K (RTK) robustness* (Section 7.2, Definition 7.2). Moreover, we demonstrate how a neural network can be instrumented (following an analogous approach to the one presented in Chapter 6) in order to naturally incorporate certifiable RTK robustness into its learning objective, making runtime certification of this property essentially free—a construction we call *RTK GloRo Nets*.

Recently, Jia et al. [68] also proposed a definition that aims to capture certified robustness for top-k predictions. However, their work differs from ours in two key ways. First, their certification method is derived from Randomized Smoothing [26, 80], which gives a stochastic guarantee and relies on hundreds of thousands of samples for conclusive certification. More importantly, Jia et al. evaluate their proposed robustness property with respect to the *ground truth* class of the point in question, making it unclear how one might certify this property on unlabeled points, e.g., those seen by the model in deployment. We provide more discussion on this problem and how our work addresses it in Appendix B.1. As we will see, devising a robustness analogue to top-k accuracy in a manner that relaxes standard robustness and does not depend on the ground truth is a subtle task—we address this issue carefully in Section 7.2.

In addition to applying to only top-1 predictions, standard robustness also focuses specifically on the problem of *undirected* adversarial examples. More concretely, adversarial examples are obtained broadly via two categories of attacks: (1) *evasion* (undirected) attacks, and (2) *targeted* (directed) attacks. Local robustness provides guarantees against the former, while the latter, which requires only a weaker guarantee, has remained largely unexplored by work on certifiable robustness [49]. In this work we introduce *affinity robustness* (Sec-

[1] We note that if it is possible for an image to contain multiple subjects—and especially if this is common in the target distribution—this may be an indicator that classification is a fundamentally ill-suited objective. Here, we propose that a robust analogue to object flagging may be the most reasonable alternative, and to our knowledge this has not been previously proposed or studied. However, our analysis in the remainder of this chapter will assume that the learner is constrained to the general objective of classification.

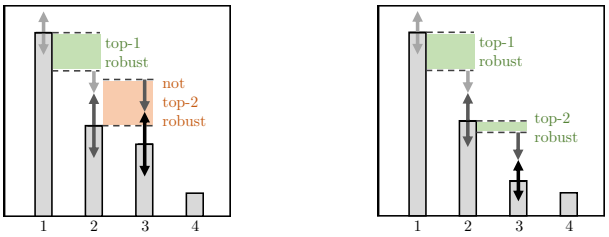


Figure 7.1: (a) Example of network outputs that demonstrate top-k robustness is not a relaxation of standard local robustness. The gray arrows denote bounds on the amount each logit can change within a radius of ϵ. We see that these bounds are not sufficient for class 2 to surpass class 1; however, the bounds are sufficient for class 3 to surpass class 2. Therefore, the point in this example is top-1 robust, but *not* top-2 robust. **(b)** Example of network outputs that are simultaneously top-1 robust and top-2 robust.

tion 7.3, Definition 7.3), which captures resistance to specified sets of directed adversarial attacks. We further show that certifiable affinity robustness can also be achieved via an analogous *Affinity GloRo Net* construction.

7.2 Relaxed Top-K Robustness

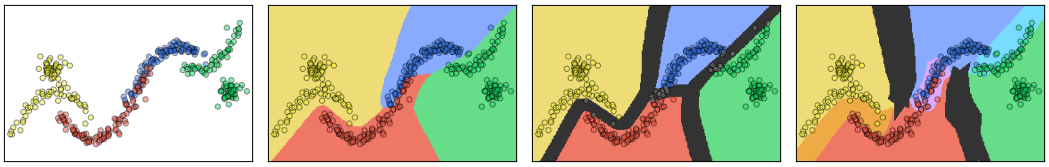
We begin in this section by introducing a notion of robustness that is inspired by the relaxed accuracy metric, top-k accuracy. Top-k accuracy is a common benchmark metric in vision applications such as Imagenet, where the class labels are particularly fine-grained, and may even be arbitrary on some instances. Furthermore, top-k accuracy has been studied as a learning objective in its own right [10], and has been identified as desirable in the context of robustness certification [68].

In order to create a notion of relaxed robustness that is analogous to top-k accuracy, we will consider the network to output a *set* of classes rather than a single class. Let us define the following notation representing the set of the top k outputs of a model: let f be the function computing the logit values of a neural network, and let $f^k(x)$ be the k^{th}-highest logit output of f on x. We then define $F^k(x) = \{j : f_j(x) \geqslant f^k(x)\}$, that is, F^k is the set of classes corresponding to the top k outputs of f. Using this notation, we define *top-k robustness* (Definition 7.1), which requires that the set of classes with the k highest logits remains invariant over small-norm perturbations. We note that if we had a single class of interest, c, e.g., the ground truth, we could simply require that c remain in F^k under small perturbations [68]; however, top-k accuracy is useful precisely because *any* of the classes in F^k could be correct, meaning that all classes in F^kshould be treated equally and guarded against perturbations.

Definition 7.1 (Top-k Robustness)**.** *A model, F, is top-k ϵ-locally-robust at point, x, w.r.t. norm, $||\cdot||$, if*

$$\forall x' \quad . \quad ||x - x'|| \leqslant \epsilon \implies F^k(x) = F^k(x')$$

While top-k robustness may appear to capture the idea of top-k accuracy, we observe that the analogy fails, as top-k robustness is *not a relaxation* of standard local robustness. For example, if a model is top-1 *accurate*, it is also top-2 accurate; however, if a model is



(a) (b) (c) (d)

Figure 7.2: (a) An example 2D synthetic dataset containing four classes. **(b)** Decision boundary of a standard model trained on the synthetic dataset. **(c)** Decision boundary of a GloRo Net [88] trained to be certifiably robust on the synthetic dataset. **(d)** Decision boundary of an RT2 GloRo Net (see Algorithm 7.1) trained to be RT2 robust on the synthetic dataset. We observe that the RT2 GloRo Net can label the points as accurately as the standard model, while the GloRo Net must reject some point on the manifold. The RT2 GloRo Net reports a relaxed robustness guarantee (indicated by orange, purple, and cyan) in the regions where the classes overlap. E.g., in the orange region, the RT2 GloRo Net guarantees that no adversary can change the label to blue or green with a small-norm perturbation.

top-1 *robust*, it may not be top-2 robust. Figure 7.1a provides an example illustrating this point.

We thus turn our attention to a modification of Definition 7.1 that *does* relax standard local robustness. Definition 7.2 provides a notion of what we call *relaxed* top-K robustness, or RTK robustness, which is properly analogous to the concept of top-k accuracy. Essentially, a point is considered RTK robust if it is top-k robust for some k in $\{1, \ldots, K\}$.

Definition 7.2 (Relaxed Top-K Robustness)**.** *A model, F , is relaxed-top-K (RTK) ϵ-locally-robust at point, x, w.r.t. norm, $||\cdot||$, if*

$$\forall x' \;\; . \;\; ||x - x'|| \leqslant \epsilon \implies \exists \; k \leqslant K : F^k(x) = F^k(x')$$

From the definition, it is clear that RTK robustness is a relaxation of standard local robustness: first, RT1 robustness is equivalent to top-1 robustness, which is equivalent to local robustness; second, RT1 robustness implies RTK robustness for $K > 1$.

We can think of RTK robustness as allowing the model to output a set of labels (with size at most K) such that the output set remains invariant under bounded-norm perturbations. In some cases there may be multiple such sets, e.g., if the model is both top-1 robust and top-2 robust on the point (see Figure 7.1b for an example). These can be thought of as the sets of classes that are "safe" to predict on a given point; if the only such set is empty then no classes are safe, and the model abstains from predicting. Geometrically, this gives rise to a spatially-arranged hierarchy of classes that is conveyed through the robustness guarantee: each region on the decision surface corresponds to some set (or sets) of classes that can be robustly predicted. Figure 7.2c gives an example of how such a decision surface might look on a synthetic 2D dataset, via a coloring that indicates the *smallest* safe set that can be predicted.

Certifying RTK Robustness

We now describe how to produce networks that incorporate certifiable RTK robustness into their training objectives. In particular we follow a similar approach to Leino et al. [88], instrumenting the output of a neural network to return an added class, $\bot$, in cases where the

Algorithm 7.1: RTK GloRo Net Prediction

Inputs: a neural network, $f : \mathbb{R}^n \to \mathbb{R}^m$; a number, $K \in \mathbb{N}$, specifying the maximum robust set size; a radius, ϵ; a point $x \in \mathbb{R}^n$

Output: a prediction $y \in [m]$ if f is ϵ-RTK-robust at x, or $\perp$ otherwise

def `rtkPredict` *(f , K , ϵ , x)*:
- $y \leftarrow f(x)$
- **for** $k \in [K]$ **do**
  - `// F^k is the set of top-k indices of y.`
  - $F^k \leftarrow \{i : y_i \geqslant f^k(x)\}$
  - **for** $j \in F^k$ **do**
    - `// m^k_j is the margin (under ε perturbation) between class j and any`
    - `// class not in the top k.`
    - $m^k_j \leftarrow y_j - \max_{i \notin F^k} \{y_i + \epsilon \cdot \texttt{lipschitzConstant}(f_j - f_i)\}$
  - `// m^k is the margin by which every class in the top k classes exceeds`
  - `// every class not in the top k.`
  - $m^k \leftarrow \min_j \in F^k\{m^k_j\}$
- **if** $\exists k \in [K] : m^k > 0$ **then**
  - **return** $\text{argmax}_i\{y_i\}$
- **else**
  - **return** $\perp$

desired property cannot be certified. Perhaps surprisingly, our construction demonstrates that our proposed robustness properties can be certified efficiently, *with little overhead compared to a forward pass of the network.*

Recall from Chapter 6 that a GloRo Net encodes robustness certification into its architecture such that all points are either rejected (the network predicts an added class, $\perp$), or certifiably ϵ-locally-robust by making use of the underlying network's global Lipschitz constant. Specifically, let f be a neural network, let j be the class predicted by f on point, x, and let K_{ji} (for $i \neq j$) be the Lipschitz constant of $f_j - f_i$. A GloRo Net adds an extra logit value, $f_\perp(x) = \max_{i \neq j}\{f_i(x) + \epsilon K_{ji}\}$. If the margin between the highest logit output, $f_j(x)$, and the second-highest logit output, $f_i(x)$, is smaller than ϵK_{ji}, then $f_\perp(x)$ will be the maximal logit score in the GloRo Net, thus the point will be rejected; otherwise the GloRo Net will not predict $\perp$ and the point can be certified as ϵ-locally-robust.

We propose a variation of GloRo Nets, *RTK GloRo Nets*, which naturally satisfy RTK robustness on all non-rejected points. As with standard GloRo Nets, we construct RTK GloRo Nets by instrumenting a model, f, such that the instrumented model returns $\perp$ unless f can be certified as RTK ϵ-locally-robust. This construction is given by Algorithm 7.1, and described below.

Let F be the predictions made by f, i.e., $F(x) = \text{argmax}_i\{f_i(x)\}$, and let $F^k(x)$ be the set of the top k predictions made by f on x; and as above, let K_{ji} be the Lipschitz constant of $f_j - f_i$.

For $k \leqslant K$ and $j \in F^k(x)$, let $m^k_j(x) = f_j(x) - \max_{i \notin F^k(x)}\{f_i(x) + \epsilon K_{ji}\}$. Intuitively, $m^k_j(x)$ is the margin by which class j, which is in the top k classes, exceeds every class not in the top k classes, after accounting for the maximum change in logit values within a radius of ϵ determined by the Lipschitz constant. We observe that if $m^k_j(x) > 0$ then the logit for

class j, $f_j(x)$, cannot be surpassed within the ϵ-ball around x by an output not in the top k outputs, $f_i(x)$ for $i \notin F^k(x)$.

Next, let $m^k(x) = \min_{j \in F^k(x)}\{m_j^k(x)\}$. This represents the minimum margin by which *every* class in the top k classes exceeds every class not in the top k classes; thus if $m^k(x) > 0$, then F is top-k robust at x.

Finally, let $m(x) = \max_{k \leqslant K}\{m^k(x)\}$. We observe that if $m(x) > 0$, then the model is RTK robust at x. We would thus like to predict $\bot$ only when $m(x) < 0$. To accomplish this we create an instrumented model, g, as given by Equation 7.1.[2] This instrumented model naturally satisfies RTK robustness, as stated by Theorem 7.1.

$$g_i(x) = f_i(x); \quad g_\bot(x) = \max_i\{f_i(x) - m(x)\} \tag{7.1}$$

Theorem 7.1. *Let g be an RTK GloRo Net as defined by Equation 7.1. Then, if the maximal output of $g(x)$ is not $\bot$, then F is RTK ϵ-locally-robust at x.*

The proof of Theorem 7.1, follows similarly to the proof of Theorem 6.1 in Chapter 6, and is given in Appendix B.2.

Other Certification Techniques. While many certification methods have been proposed and studied in the past, we use the GloRo Net approach to certification for two key reasons. First, as we have seen, GloRo Nets are among the top state-of-the-art methods for deterministic ℓ_2 certification. Second, GloRo Nets provide an elegant way for incorporating our robustness objectives into a *certifiable training procedure*, as certified training can be reduced to simply performing overapproximate certification. However, leveraging other types of techniques for certifying our proposed robustness properties remains an interesting possibility for future work to explore. Appendix B.3 provides a discussion of how other certification techniques may be adapted to incorporate RTK and affinity robustness.

7.3 Affinity Robustness

In Section 7.2 we demonstrated how to relax local robustness in order to capture a robustness notion analogous to the concept of top-k accuracy. This induces a decision surface with a spatially-arranged hierarchy of classes conveyed through the robustness guarantee (see Figure 7.2d). While logical hierarchies may arise naturally, in some cases we may wish to guide the hierarchy that forms. E.g., we may have *a priori* knowledge of the class hierarchy that we would like to impart to our model, or we may have a limited set of class groupings that are acceptable.

In this section, we demonstrate how the class hierarchy induced on the decision surface of a model can be guided using pre-specified *affinity sets*, which define the sets of classes that can be grouped together without intervening rejected space on the decision surface. More formally, let $\mathcal{S}$ be a collection of affinity sets, where each affinity set $S \in \mathcal{S}$ is simply a set of class labels. We will typically assume that for each class, j, $\exists\ S \in \mathcal{S}$ such that $j \in S$; that is, each class is included in at least one affinity set.

For a given collection of affinity sets, we can define *affinity robustness* (Definition 7.3), which stipulates that points, x, may be top-k robust for any k, so long as $F^k(x)$ is contained

[2] When g outputs $\bot$, we want the gradient of Equation 7.1 to encourage increasing $m(x)$, but *not* encourage *decreasing* $\max_i\{f_i(x)\}$. Thus we treat f_i as a constant in Equation 7.1, such that gradient information only flows through m.

in some affinity set. That is, classes may be "grouped" together in a robust prediction set with other classes that share an affinity set.

Definition 7.3 (Affinity Robustness)**.** *A model, F, is affinity-ϵ-locally-robust on a point, x, w.r.t. a collection of affinity sets, $\mathcal{S}$, and norm $||\cdot||$, if*

$$\forall x' \;.\; ||x - x'|| \leqslant \epsilon \implies \exists\; k \in \mathbb{N},\; S \in \mathcal{S} \;:\; F^k(x) = F^k(x') \quad \wedge \quad F^k(x) \subseteq S$$

Examples of Affinity Sets. Several common datasets offer natural instantiations of affinity sets. For example, CIFAR-100 contains 100 classes that are grouped into 20 super-classes containing 5 related classes each. Additionally, Imagenet classes belong to a tree structure from which many natural collections of affinity sets could be derived. Finally, affinity sets can be used to capture mistakes that may arise even without adversarial manipulation, e.g., by grouping classes that are visually similar; or to group classes that may naturally co-occur in the same instance, e.g., highways and pastures in EuroSAT [59] satellite images (see Section 7.4.2).

Certifying Affinity Robustness

We now describe how to produce networks that incorporate certifiable affinity robustness into their training objectives. As with the construction for RTK GloRo Nets in Section 7.2, we again instrument the output of an underlying neural network to return an added class, $\bot$, in cases where affinity robustness cannot be certified. We call this construction *Affinity GloRo Nets*, which naturally satisfy affinity robustness on all non-rejected points. Our construction is given by Algorithm 7.2, and described below.

Again, we let F be the predictions made by f, i.e., $F(x) = \text{argmax}_i\{f_i(x)\}$; we let $F^k(x)$ be the set of the top k predictions made by f on x; and we let K_{ji} be the Lipschitz constant of $f_j - f_i$. Furthermore, let $\mathcal{S}$ be a collection of affinity sets, and let $K = \max_{S\in\mathcal{S}}\{|S|\}$; that is, K is the size of the largest affinity set in $\mathcal{S}$.

As before, we define $m_j^k(x)$ for $k \leqslant K$ as $m_j^k(x) = f_j(x) - \max_{i\notin F^k(x)}\{f_i(x) + \epsilon K_{ji}\}$, and $m^k(x)$ for $k \leqslant K$ as $m^k(x) = \min_{j\in F^k(x)}\{m_j^k(x)\}$, where $m_j^k(x)$ represents the margin by which class j, which is in the top k classes, exceeds every class not in the top k classes, after accounting for the maximum change in logit values within a radius of ϵ determined by the Lipschitz constant, and represents the minimum margin by which *every* class in the top k classes exceeds every class not in the top k classes.

Now, let $m(\mathcal{S}, x)$ be given by Equation 7.2. Essentially, we restrict our consideration of sets of top-k predictions to those that are constrained to a single affinity set. Among the considered sets, we take the maximum margin by which every class in the set will surpass every class not in the set under bounded perturbations to x.

$$m(\mathcal{S}, x) = \max_{k \;:\; \exists S\in\mathcal{S} \;:\; F^k(x)\subseteq S} \left\{\; m^k(x) \;\right\} \tag{7.2}$$

We note that in practice, the maximum in Equation 7.2 can be computed efficiently by representing the sets $F^k(x)$ and S as bit maps and masking out rows of m^k that correspond to values of k for which $F^k(x) \cap S \neq F^k(x)$ for all $S \in \mathcal{S}$. This is reflected in our implementation, which can be found on GitHub.[3]

[3] Code available at `https://github.com/klasleino/gloro`.

Algorithm 7.2: Affinity GloRo Net Prediction

Inputs: a neural network, $f : \mathbb{R}^n \to \mathbb{R}^m$; a collection of affinity sets, $\mathcal{S}$; a radius, ϵ; a point $x \in \mathbb{R}^n$

Output: a prediction $y \in [m]$ if f is ϵ-Affinity-robust w.r.t. $\mathcal{S}$ at x, or $\perp$ otherwise

```
def rtkPredict (f , S , ε , x):
    y ← f(x)
    // K is the set of values of k for which the top k logits in y belong to a
    // single affinity set.
    K ← {}
    for k ∈ [max_{S∈S}{|S|}] do
        // F^k is the set of top-k indices of y.
        F^k ← {i : y_i ⩾ f^k(x)}
        for S ∈ S do
            if F^k ⊆ S then
                K ← K ∪ {k}
    for k ∈ K do
        F^k ← {i : y_i ⩾ f^k(x)}
        for j ∈ F^k do
            // m^k_j is the margin (under ε perturbation) between class j and any
            // class not in the top k.
            m^k_j ← y_j − max_{i∉F^k} {y_i + ε · lipschitzConstant(f_j − f_i)}
        // m^k is the margin by which every class in the top k classes exceeds
        // every class not in the top k.
        m^k ← min_j ∈ F^k {m^k_j}
    if ∃k ∈ K : m^k > 0 then
        return argmax_i{y_i}
    else
        return ⊥
```

We observe that if $m(\mathcal{S}, x) > 0$, then the model is affinity robust at x. We would thus like to predict $\perp$ only when $m(\mathcal{S}, x) < 0$. To accomplish this we create an instrumented model, g, as given by Equation 7.3. The correctness of this approach is stated in Theorem 7.2, which we prove in Appendix B.4.

$$g_i(x) = f_i(x); \quad g_\perp(x) = \max_i \{f_i(x) - m(\mathcal{S}, x)\} \tag{7.3}$$

Theorem 7.2. *Let g be an Affinity GloRo Net as defined by Equation 7.3. Then, if the maximal output of $g(x)$ is not $\perp$, then F is affinity-ϵ-locally-robust at x.*

7.4 Relaxed Robustness in Practice

In this section, we motivate our proposed robustness relaxations via an empirical demonstration and argue that our proposed relaxations are likely to be relevant for extending certified defenses to complex prediction tasks with many classes, where standard robustness may be difficult or unrealistic to achieve. To this end, we first find that applying these relaxed

robustness variants to suitable domains leads to certifiable models with lower rejection rates and higher certified accuracy (Section 7.4.1). We then explore the intriguing properties of a model's prediction space that arise when the model is trained with the objective of RTK or affinity robustness on appropriate domains. In particular, we examine the predictions of RTK and Affinity GloRo Nets trained on EuroSAT [59] (Section 7.4.2) and CIFAR-100 (Section 7.4.3), and find that (1) the classes that are "grouped" by the model typically follow a logical structure, even without supervision, and (2), affinity robustness can be used to capture a small set of specific, challenging class distinctions that account for a relatively large fraction of the points satisfying RTK robustness but not standard robustness.

In addition, we provide further motivation for Affinity GloRo Nets in Section 7.4.4 by showing how they can be leveraged to efficiently certify a previously-studied safety property for ACAS Xu [70], a collision avoidance system for unmanned aircraft that has been a primary motivation in many prior works that study certification of neural network safety properties [49, 71, 72, 87, 95]. Specifically, we show that Affinity GloRo Nets can certify *targeted safe regions* [49] in a single forward pass of the network, while previous techniques based on formal methods require hours to certify this property, even on smaller networks [49].

7.4.1 Improving Certified Performance through Relaxation

We begin our evaluation by measuring the extent to which certification performance can be improved when the objective is relaxed. We focus particularly on the ℓ_2 norm, and *deterministic* certification and guarantees, as opposed to the types of guarantees obtained via Randomized Smoothing [26]. To this end, we compare against GloRo Nets [88], which have achieved state-of-the art performance for deterministic certification on several common benchmark datasets.

Experimental Setup

We begin by briefly describing the setup used for the experiments in this section.

Datasets. Our evaluation focuses on datasets for which our relaxed robustness variants are appropriate. Namely, we select datasets with large numbers of fine-grain classes, or classes with a large degree of feature-overlap: EuroSAT [59], CIFAR-100 [76], and Tiny-Imagenet [78]. The most widely-studied datasets for benchmarking deterministic robustness certification are CIFAR-10 and MNIST, which do not fit these desiderata; however, Tiny-Imagenet has also been used for evaluating deterministic certification [82, 88], making our results directly comparable to the previously-published state-of-the-art.

Models. We trained three types of models in our evaluation: GloRo Nets (as a point of comparison to standard robustness certification), RTK GloRo Nets, and Affinity GloRo Nets. More details on the affinity sets chosen for the Affinity GloRo Nets are given in Section 7.4.2 (for EuroSAT) and Section 7.4.3 (for CIFAR-100). The details of the architecture, training procedure, and hyperparameters are provided in Appendix B.5.

Metrics. We measure the performance of RTK models using a metric we call *RTK VRA*, which is the natural analogous metric to top-k accuracy. For a given point, x, that is certifiably RTK robust, let k^* be the the maximum $k \leqslant K$ such that the model is top-k robust

dataset	*guarantee*	ϵ	*VRA / RTK VRA / affinity VRA*	*rejection rate*	*clean accuracy**
EuroSAT	standard (local robustness)	0.141	0.749 ± 0.003	0.204 ± 0.002	0.862 ± 0.003
EuroSAT	RT3	0.141	0.908 ± 0.002	0.073 ± 0.002	0.987 ± 0.002
EuroSAT	highway+river affinity	0.141	0.798 ± 0.002	0.170 ± 0.002	0.917 ± 0.003
EuroSAT	highway+river+agriculture affty.	0.141	0.819 ± 0.003	0.151 ± 0.003	0.930 ± 0.003
CIFAR-100	standard	0.141	0.281 ± 0.002	0.640 ± 0.003	0.473 ± 0.002
CIFAR-100	RT5	0.141	0.360 ± 0.002	0.562 ± 0.002	0.706 ± 0.003
CIFAR-100	superclass affinity	0.141	0.323 ± 0.002	0.599 ± 0.002	0.520 ± 0.002
Tiny-Imagenet	standard	0.141	0.210[88]	0.639[88]	0.346[88]
Tiny-Imagenet	RT5	0.141	0.277 ± 0.002	0.447 ± 0.006	0.537 ± 0.002
ACAS Xu	targeted affinity	0.010	0.749 ± 0.001	0.195 ± 0.001	0.858 ± 0.002

Table 7.3: Certification results under various notions of robustness. Note that VRA and clean accuracy numbers are given for the VRA/accuracy metric corresponding to the robustness guarantee the respective model was trained for, thus their meaning is slightly slightly different for each guarantee (see Section 7.4.1:**Metrics**). Results are taken as the average over 10 runs; standard deviations are denoted by ±.

at x (recall that an RTK robust point can be top-k robust for more than one k—k^* corresponds to the *loosest* such guarantee). We define the RTK VRA of model, F, as the fraction of labeled points, (x, y), such that (1) the model is RTK robust at x, and (2) $y \in F^{k^*}$. In other words, the correct label must be in a *certifiably-robust set of top-k predictions* for some $k \leqslant K$. Similarly, we define *affinity VRA*, with respect to a collection of affinity sets, $\mathcal{S}$, as the fraction of points for which the correct label is in a certifiably-robust set of top-k predictions that is contained in some affinity set in $\mathcal{S}$.

We also provide clean accuracy metrics for each model, corresponding to the guarantee the model is trained for; e.g., top-k accuracy for RTK GloRo Nets. For Affinity GloRo Nets, we use what we call *affinity accuracy*, which counts a prediction as correct if all labels scored above the ground truth share a single affinity set with the ground truth.

Evaluation Results

Table 7.3 shows the performance of GloRo Nets compared to RTK GloRo Nets and affinity GloRo Nets with respect to the appropriate VRA metric, as well as the rejection rate of each model, i.e., the fraction of points that cannot be certified. RTK GloRo Nets use the most relaxed objective, and accordingly, we see that they consistently outperform the standard GloRo Net, improving VRA performance by 6-16 percentage points. Additionally, RTK GloRo Nets reduce the rejection rate significantly, rejecting as few as half the number of points rejected by the standard GloRo Nets. We also observe that affinity GloRo Nets consistently improve performance compared to standard GloRo Nets. In particular, highway+river+agriculture affinity (see Section 7.4.2) and superclass affinity (see Section 7.4.3) increase VRA performance by 8 points and 4 points respectively, despite the fact that these affinity guarantees are significantly more restrictive than the RTK guarantees.

Relaxed Guarantees & Learning Objectives. As demonstrated by the results in Table 7.3, RTK and affinity robustness significantly improve certifiability and VRA performance. Clearly, this improvement is in part due to the relaxed certification and evaluation criteria entailed by RTK and affinity robustness. However, there is also evidence that the relaxed learning objective itself may better aid in learning robust boundaries in general.

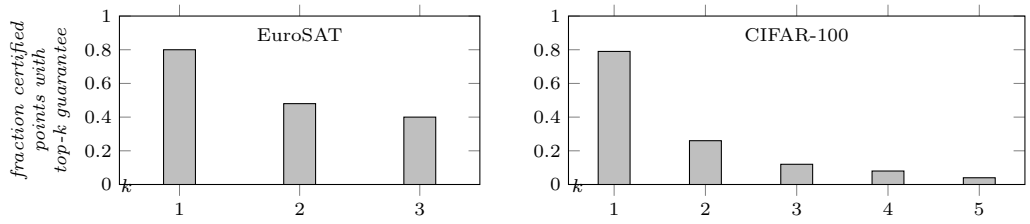


Figure 7.4: Robust prediction set sizes on EuroSAT (left) and CIFAR-100 (right). The y-axis measures the fraction of certified test instances that were certifiable as top-k robust, for each k on the x-axis. Note that the fractions do not sum to 1 because some points can receive multiple guarantees.

For example, on CIFAR-100, 5% of points rejected by the GloRo Net are certified by the RT5 GloRo Net with a *top-1 guarantee*, suggesting that the RT5 objective better facilitated obtaining a strong robustness guarantee on these points.

Similarly, we find that the objective of affinity robustness, while technically stronger than RTK robustness, guides the model towards greater certifiability than would be explained by the fraction of instances for which the RTK GloRo Net naturally satisfies the affinity set groupings, highlighting the significance of incorporating affinity robustness into the training objective. For example, on CIFAR-100, 22% of points certified by the RT5 GloRo Net receive a top-k guarantee that does not correspond to a superclass. This would correspond to a 16% higher rejection rate under superclass affinity robustness; meanwhile the rejection rate of the superclass Affinity GloRo Net is in fact only 6% higher than that of the RT5 GloRo Net.

Top-k Prediction Set Sizes. While appropriate relaxations are useful in many contexts, it is generally preferable to be able to obtain a *stricter* guarantee—i.e., a *smaller* robust output set. Figure 7.4 shows the robust prediction set sizes obtained on certified (i.e., non-rejected) test points for both EuroSAT and CIFAR-100. Note that points may receive a top-k guarantee for multiple values of k, meaning that a point may belong to robust sets of more than one size. For example, 56% of certified points on EuroSAT and 24% on CIFAR-100 received a guarantee for multiple values of k. We see that the majority of points get a tight guarantee, with about 80% of certified points receiving a top-1 guarantee, and looser guarantees becoming progressively less common.

7.4.2 Relaxed Robustness Guarantees on EuroSAT

EuroSAT is a relatively recent dataset based on land-use classification of Sentinel-2 satellite images [59]. Although EuroSAT contains only ten classes, we argue that it is nonetheless a suitable application for the types of relaxed robustness presented in this paper, primarily because of the lack of mutual exclusivity among its classes. Specifically, the classification task proposed for EuroSAT contains the following classes to describe a 64×64 image patch: (1) annual crop, (2) forest, (3) herbaceous vegetation, (4) highway, (5) industrial buildings, (6) pasture, (7) permanent crop, (8) residential buildings (9) river, and (10) sea/lake. Each instance has exactly one label, however, in practice the labels are not necessarily non-overlapping. For example, highway images may depict a road going through agricultural land, crossing a river, or near buildings. It may be reasonable, then, for a classifier to produce

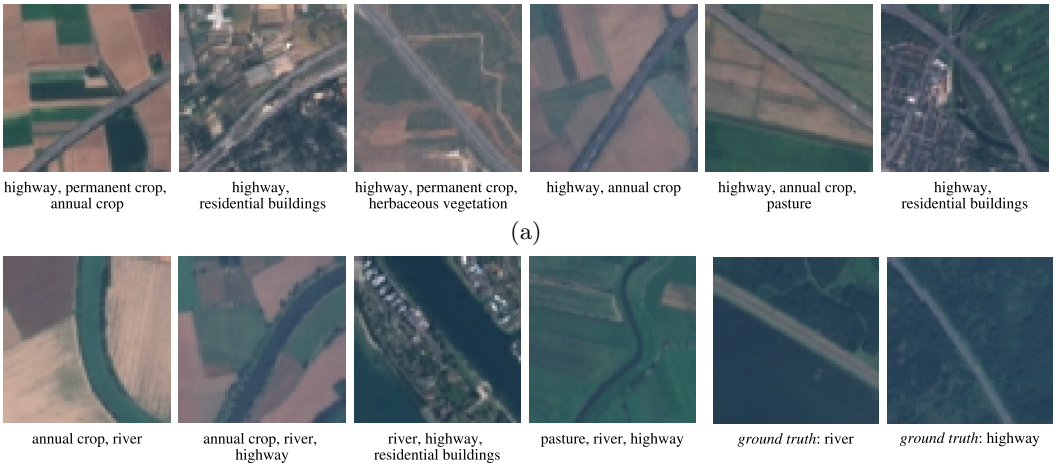


Figure 7.5: Samples of EuroSAT instances labeled "highway" **(a)** or "river" **(b)** that are rejected (i.e., cannot be certified) by a standard GloRo Net, but certified by an RT3 GloRo Net. The classes included in the RT3 robustness guarantee are given beneath each image. **(c)** Two visually similar instances with ground truth label "river" (left) and "highway" (right).

high logit values for two classes simultaneously, making local robustness potentially difficult to achieve.

Indeed, we find this to be the case; namely, many EuroSAT instances labeled "highway" cannot be certified for standard robustness, while a large fraction of these rejected inputs *can* be certified for RT3 robustness. Specifically, we found that a standard GloRo Net trained on EuroSAT rejected 39% of highway instances from the test set (above average for instances overall). Meanwhile, an RT3 GloRo Net was able to certify 72% of these same instances. Upon examination, we found that many of the guarantees associated with the points rejected by the GloRo Net but certified by the RT3 GloRo Net appeal nicely to intuition. Figure 7.5a depicts a few of these intuitive examples; we see that on image patches with a highway in a field, the RT3 GloRo Net gives top-2 or top-3 guarantees with highway alongside classes such as "annual crop," and on image patches with a highway near a neighborhood, it gives top-2 guarantees with highway grouped with "residential buildings."

We see a similar trend for the "river" class, shown in Figure 7.5b. Additionally, we find that many instances of rivers receive an RTK guarantee including "highway" as one of the classes belonging to the corresponding robust prediction set. As illustrated by Figure 7.5c, this may not be unreasonable—Figure 7.5c shows two visually similar EuroSAT instances with ground truth label "river" and "highway" respectively, demonstrating that the 64×64 patches may not always provide enough detail and context to easily distinguish these two classes.

The above intuition suggests that the difficulty in learning a certifiably-robust model on EuroSAT may be largely due to frequent cases where specific sets of classes may not be sufficiently separable. That an adversary might have the ability to control which of a set of plausible labels is chosen may be considered inconsequential, provided the adversary cannot cause *arbitrary* mistakes. This observation motivates the use of affinity robustness on EuroSAT. That is, we may wish to further restrict the sets of classes that may forgo ϵ-

Figure 7.6: Samples of CIFAR-100 instances that are both correctly classified and certified as top-k robust for $k > 1$. The classes included in the RT5 robustness guarantee are given beneath each image.

Figure 7.7: Comparison of robust prediction sets produced by an RT5 GloRo Net (left) and a superclass Affinity GloRo Net (right). Samples are taken from points on which the RT5 GloRo Net did not match a single superclass, while the Affinity GloRo Net was able to successfully certify the point.

separation (those that correspond to "arbitrary" mistakes), while at the same time admitting the model to group a specified set of classes, between which adversarial examples could be considered benign.

To this end, we suggest two plausible affinity sets for EuroSAT. The first, which we refer to as *highway+river affinity*, captures the challenges faced by standard GloRo Nets that are illustrated by Figure 7.5; namely, $\mathcal{S}$ consists of one affinity set, S_c per class, c, consisting of c, the class "highway," and the class "river." The second, which we refer to as *highway+river+agriculture affinity* additionally allows the classes "permanent crop" and "annual crop" to be grouped, as these classes are often visually similar. We find that these two affinity sets allow us to improve the VRA on EuroSAT compared to a standard GloRo Net by 5 and 7 percentage points, respectively (see Table 7.3). Moreover, the performance of the highway+river+agriculture Affinity GloRo Net closes half of the VRA gap between the standard GloRo Net and the more-relaxed RTK GloRo Net, suggesting that roughly half of the performance benefits obtained under RTK robustness can be recovered by accounting for a few simple types of reasonable mistakes.

7.4.3 Relaxed Robustness Guarantees on CIFAR-100

CIFAR-100 is another natural application for which our relaxed robustness variants are suitable for several reasons, including that (1) it contains a large number of classes, (2) many of the classes are fine-grain, especially considering the small size of the input images, and (3) its 100 classes are further organized into 20 *superclasses* that each encompass 5 classes.

We find that when training for RT5 robustness using an RT5 GloRo Net, 78% of certifiable points on the RT5 GloRo Net have corresponding robust prediction sets that are contained by some superclass set. That is, even without supervision, the robust prediction

sets of the RTK GloRo net typically respect the superclass hierarchy. Moreover, even on instances for which the robust prediction set does not match a superclass set, the robust prediction set is often nonetheless "reasonable," in that it is often clear upon inspection why the model may have chosen the particular set of predictions. Figure 7.6 provides samples of correctly-classified, RT5-certifiable points with their corresponding robust prediction sets, illustrating this point. More such examples can be found in Appendix B.6.

However, supposing we would want to strictly enforce the model's robust prediction sets to be contained entirely in one superclass, Affinity GloRo Nets provide a means of doing this. We find that 34% of instances on which the RT5 GloRo Net fails to match a superclass can be certified by the superclass Affinity GloRo Net. Figure 7.7 provides examples of such instances, showing that the additional supervision of Affinity GloRo Nets helps better ensure the robust prediction sets respect superclasses.

7.4.4 Targeted Safe Regions on ACAS Xu

ACAS Xu is an airborne collision avoidance system for unmanned aircraft that has been studied in the context of neural network safety certification [49, 71, 72, 87, 95]. The classification task for ACAS Xu is as follows. Given a few features, e.g., altitude, velocity, etc., the network produces a horizontal maneuver advisory for the aircraft, instructing it on how to turn to avoid a collision. This advisory comes from one of the following options: (1) hard left, (2) left, (3) clear of conflict (i.e., go straight), (4) right, or (5) hard right.

Access to the ACAS Xu dataset is not public. However, many trained networks that have been certified for other safety properties specific to ACAS Xu (and unrelated to robustness) are publicly available; we generated a synthetic dataset derived from the predictions of one such public model provided by Katz et al. [71]. To create this dataset, we generated random inputs clipped to be within the standard range for each input [71] and labeled them using a publicly-available pretrained ACAS Xu network.

We chose our value for ϵ by estimating the minimum ℓ_2 distance between any two points with different labels. This value was approximately 0.02 on our synthetic dataset; thus we used $\epsilon = 0.01$, as ϵ-local-robustness requires a separation of 2ϵ between classes.

Previously, Gopinath et al. [49] proposed certification of *targeted safe regions* on ACAS Xu. An ℓ_p ball with radius ϵ, centered at x, is considered *targeted safe* if the horizontal maneuver advisory does not change within the ball except to either one degree further left or one degree further right. E.g., an ℓ_p ball may contain the directives "left" and "hard left" or "left" and "clear of conflict." We note that this property can be captured by affinity robustness, where the affinity sets are simply all pairs of adjacent directives.

Table 7.3 presents the results of training and evaluating an Affinity GloRo Net on our synthetic ACAS Xu dataset. In particular, we give the VRA (as the fraction of points that are correctly classified and affinity robust), the rejection rate (as the fraction of points that cannot be certified as affinity robust), and the clean accuracy. Gopinath et al. do not present any directly-comparable metrics to these in their evaluation. Rather than presenting rejection rates for a fixed epsilon, they instead attempt to determine the minimum radius under which the property holds, and they present the average such radius. It is therefore unclear what fraction of points would be accepted under any fixed radius. However in Gopinath et al.'s experiments, 16% of the points timed out after 12 hours, meaning that *at least* 16% of points were unable to be certified. This is comparable to the 19% rejection rate obtained by our Affinity GloRo Net. Moreover, the points that were successfully certified by Gopinath et al. took, on average, 7.6 hours to certify; by comparison, our approach certifies points in

a single forward pass of the network, meaning *an entire batch* can be certified *in a matter of milliseconds.*

7.5 Robustness Does Not Imply Conceptual Soundness

Local robustness is an important property that captures resistance specifically to small-norm adversarial examples. The problem of guarding against this class of adversarial examples is so widely studied that local robustness is often informally referred to as simply "robustness." While ℓ_p-bounded perturbations are intended to capture a subset of *imperceptible* perturbations—the initial description of the adversarial perturbations discovered by Szegedy et al. [141]—we should not be deceived us into thinking that "robust" models—even those that are provably "robust" in this sense—are invulnerable to all forms of adversarial examples.

First, imperceptible perturbations do not necessarily have small ℓ_p norms. For example, translating the subject of a picture by one pixel may be This cannot be addressed by increasing the norm bound, as ℓ_p distances (especially ℓ_∞) do not align precisely with perceptibility—let alone semantics—thus this may admit meaningful changes [146], which should not be considered adversarial perturbations. Some prior work has considered generalizations of ℓ_p local robustness meant to better capture perceptual invariance, though not typically in the context of certification. For example, previous literature has proposed training for invariance under unions of multiple ℓ_p balls [28, 145], and invariance to rotations and translations [36].

Beyond this, adversarial examples may still be derived from perceptible perturbations that are inconspicuous and semantically meaningless to the classification task. In many ways, these types of adversarial examples pose a larger security threat, as they are more likely to be physically realizable. For example, Sharif et al. [126] designed carefully crafted patterned glasses that can be worn to fool facial recognition models into making arbitrary misclassifications. While the glasses are clearly visible, a human onlooker would not expect them to have any profound impact on the wearer's perceived identity. Other sorts of physically realizable attacks have also been proposed [16, 38, 77]. It is not clear that any easily defined geometric property characterizes such attacks, making it much more difficult to conceive of corresponding provable defenses. At a high level, we would require a sort of "semantic robustness," which demands insensitivity to the set of semantically irrelevant perturbations. For tasks involving human perception, semantic robustness takes on a Turing-test-like quality—we simply want our models to avoid mistakes that humans would be unlikely to make—a view underscoring the difficulty of this task.

Finally, conceptual soundness relates to a number of model quality issues that are not directly tied to adversarial examples; Part III discusses this further.

Part III

Conceptual Soundness Beyond Robustness

Chapter 8

Privacy Risks of Conceptual Unsoundness

We have seen the vital connection between robustness and conceptual soundness, discussed in depth in Part II. A lack of robustness to adversarial examples can be viewed as an indication that a model is overly sensitive to semantically inconsequential features; and therefore, it would be difficult to deem any model with such vulnerabilities as conceptually sound. However, as observed in the conclusion of Chapter 7, robustness and conceptual soundness are not equivalent; machine learning models may also exhibit other weaknesses that are not directly related to adversarial examples. In the final part of this thesis, we will explore and analyze some of the further defects that can be related to conceptual unsoundness. This chapter will focus on privacy vulnerabilities, and Chapter 9 will cover calibration issues.

An adversarial example, by definition, must have downstream effects on the model's ultimate prediction. But even when a reliance on conceptually unsound features is not realized this overtly, it can nonetheless compromise the integrity of the model. In this chapter, we will discuss how conceptual unsoundness can introduce privacy vulnerabilities, wherein private information about the data a model is trained on can be leaked through the model's parameters or behavior. Specifically, we show that when models use features that are overly-specific to private aspects of their training data—rather than appropriately general ones—then private information is leaked through the model.

We will begin in Section 8.1 with an overview of some of the privacy risks that have been studied in deep networks. Next, we will demonstrate how an adversary can exploit a model's lack of conceptual soundness to perform an attack called *membership inference*, in which an adversary identifies points belonging to an ostensibly private data set that was used to train a model (Sections 8.2 and 8.3). Finally, we will cover attacks that attempt to directly reconstruct private attributes of a training set—including entire training instances in some cases—in order to infer knowledge that the model would reasonably have been expected to keep confidential. Section 8.4 describes a threat called *model inversion*, and reveals how this threat interacts with robustness. Section 8.5 demonstrates how model inversion can enable an adversary to gain knowledge about private dataset properties, and Section 8.6 reveals how unsound feature encodings at the root of this vulnerability can lead to catastrophic leakage of specific private training points.

8.1 Overview of Privacy Risks in Machine Learning

Many compelling applications of machine learning involve the collection and processing of sensitive personal data, giving rise to concerns about privacy [5, 13, 27, 35, 42, 43, 83, 90, 128, 158]. In particular, when machine learning algorithms are applied to private training data, the resulting models might unwittingly leak information about that data through their behavior or representation. This fact can be exploited by an adversarial agent through a variety of privacy-compromising attacks.

8.1.1 Membership Inference

Membership inference attacks aim to determine whether a given data point was present in the training set used to build a model. This can be a privacy threat in itself, but vulnerability to membership inference has also come to be seen as a more general indicator of whether a model leaks private information [90, 93, 128, 161], and is closely related to the guarantee provided by *differential privacy* [33]—a strong, formal notion of data set privacy.

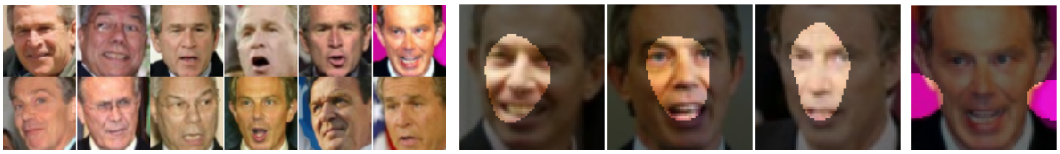
We model membership inference as a game played by an attacker, given formally by Definition 8.1 [161]. Essentially an attacker is given a point that is either drawn from the training data of a target model or from the general distribution that the training data were sampled from, each with equal probability. The attacker's goal is to predict which of the two the given point was drawn from, given access to the point and to some additional information determined by the *threat model*—we use Threat Model 8.1 below. Because an attacker that guesses randomly achieves 50% accuracy, we often refer to the *advantage* of the adversary [161], which subtracts out the 50% baseline and re-scales to lie in the range [-1, 1].

Definition 8.1 (Property Inference Game)**.** *Let $\hat{g} : \mathbb{R}^n \to \mathbb{R}^m$ be a target model that has been trained on a set $S \subset \mathbb{R}^n \times [m]$, with elements drawn i.i.d. from some underlying distribution, $\mathcal{D}^*$. Let T be a random variable drawn uniformly at random from $\{0, 1\}$. If $T = 0$, the attacker is then given a fresh instance $(x, y) \sim \mathcal{D}^*$; otherwise, if $T = 1$, (x, y) is sampled uniformly at random from the elements of the training set, S. The adversary's goal is to predict the value of T (i.e., whether (x, y) was a member of the training set or not) given (x, y) and some additional knowledge about $\hat{g}$ determined by the threat model.*

Threat Model 8.1 (Strong White-box MI Adversary)**.** *We assume that the adversary has access the following: (i) the target model function, f, including its parameters, θ (i.e., the adversary has white-box access to f); (ii) the training algorithm, $\mathcal{A}$, used to produce f, including any hyperparameters; and (iii) a set of auxiliary data sampled from $\mathcal{D}^*$ but distinct from the training set S.*

To date, most membership inference attacks follow the so-called *shadow model* approach [128]. This approach casts the attack as a supervised learning problem, where the adversary is given a data point and its true label, and aims to predict a binary label indicating membership status. To do so, the adversary trains a set of *shadow models* to replicate the functionality of the target model, and trains an *attack model* from data derived from the shadow models' outputs on the points used to train each shadow model and points not previously seen by each shadow model.

The shadow model approach uses a *black-box* threat model, where the adversary is given only oracle access to the model's outputs (including the probabilities associated with each



(a) (b) (c)

Figure 8.1: Pictorial example of how overfitting can lead to idiosyncratic use of features. **(a)** shows 12 training instances. We see that the image of Tony Blair on the top right has a distinctive pink background. **(b)** depicts internal explanations [85] for three test instances. The explanations show that the model uses Tony Blair's face to classify these instances, as we might expect. Meanwhile, **(c)** shows the explanation for the image with the distinctive pink background from the training set, where we see that the model is using the pink background to infer that the image is of Tony Blair.

class), as opposed to access to the model itself. Initially, it would seem that *white-box* information (i.e., the model's learned parameters, etc.) would provide a large edge to an attacker; while this may be true, surprisingly, prior work has not found a way to effectively leverage this information without significant relaxations to the threat model [107]. However, we show that with a deeper understanding of the mechanisms underlying overfitting, white-box information can be used to advance the state-of-the-art in membership inference (Leino and Fredrikson [83]).

Our work takes a fresh look at the problem of white-box membership inference, beginning with the intuitive observation that when a model "memorizes" certain aspects of the training data, *it is likely to show up in the way that the model uses features*—both those that are given explicitly and that are learned in internal layers. This may occur even when signs of overfitting are not clearly manifested in the model's output behavior.

Consider the example illustrated by Figure 8.1, in which a model was trained to recognize faces from the *Labeled Faces in the Wild* (LFW) dataset. Figure 8.1a shows 12 instances sampled from the training set of the model. The top right corner of Figure 8.1a depicts an image of Tony Blair with a distinctive pink background. Supposing that the background is unique to this training instance, an overfit model may use the background as a feature for classifying Tony Blair, identifying the instance as a member of the training set via the uncharacteristic way in which the model correctly labels it. In such a setting, the model's use of the pink background could be viewed as evidence of membership. Indeed, Figures 8.1b and 8.1c show this phenomenon on a convolutional neural network trained on this dataset. Figures 8.1b and 8.1c use *Internal Influence* [85] (Chapter 3) to visualize the regions of the image that are most influential towards the classification of "Tony Blair" on three test instances, and on the aforementioned training instance with the pink background. While the model is influenced most by Tony Blair's face for classification on the test instances, on the training instance it relies on the distinctive pink background.

This example highlights the intuition that *membership information is leaked via a target model's idiosyncratic use of features.* Essentially, features that are distributed differently in the training data from how they are distributed in the true distribution can provide evidence either for or against membership when the model inappropriately fixates on them. Our attack works by deriving a set of parameters that profile idiosyncratic feature use, which are then used to construct a logistic attack model. The parameters of the attack are derived to be optimal against linear models in a simplified setting with strong distributional assumptions, but the attack can be applied effectively without any such assumptions. The

success of our attack supports the primary hypothesis motivating this work, namely that *conceptual unsoundness is a privacy liability.*

8.1.2 Property Inference

Rather than attempting to isolate specific training points, an adversary might instead be interested in gaining broader knowledge about a model's training set. Property inference attacks [5, 20, 45, 102, 112] seek to infer distributional information about a model's training data given access to the model itself—whether that be in a white-box [45] or black-box [20] setting. These kinds of attacks differ subtly from attribute inference and reconstruction attacks [43, 135], which focus on inferring an attribute value of an individual instance in the training dataset. Instead, property inference attacks seek to identify an attribute value of the training dataset as a whole. Indeed, property inference, attribute inference, and to some extent reconstruction each rely on the sensitivity of the model to distributional characteristics of the training dataset—both in the sense of the underlying generative process, and in the sense of artifacts resultant from finite sampling thereof.

Property inference attacks typically consider the property of interest to be the relative proportion of a sub-population bearing some attribute—for example, the proportion of women in US census data [45]. The inference of discrete properties has also been considered, e.g., inferring the presence of individuals wearing glasses in the training data of a gender classifier [102]. However, prior work on property inference attacks typically lack control with respect to the association between the property of interest and the predictive task itself [5, 20, 45]. It is difficult to claim that a model should *not* reveal the target property in scenarios with clear association (e.g., gender and income prediction), as there is not a sufficient distinction between these properties and the premises of statistical learning in the first place.

Without direct control over the generative process by which properties occur in the training data, these concerns are difficult to navigate. As such, we contribute a novel methodology in which we constrain our analysis to properties whose generative model we explicitly control—namely the addition of watermarks to the data (Leino et al. [89]). In this way, we concretely enforce a lack of association between target properties and the predictive task, meaning that such properties have no reason to be learned by the model, and thus that their disclosure is a valid privacy concern. Interestingly, we find that *even statistically irrelevant properties may nonetheless be encoded and leaked by neural network classifiers.*

Figure 8.2 shows prototypical examples of the types of watermarks we apply to each dataset in our evaluation. Because we use watermarks to represent a class of dataset properties for which we may appropriately have an expectation of privacy, we may use the terms "watermark" and "private property" interchangeably.

We will formalize property inference in terms of boolean predicates on *training sets*, Φ, which indicate whether a dataset exhibits a particular property (e.g., if the dataset has been augmented with a particular watermark). Additionally, for our methodology, it will be useful for us to define a generative process corresponding to a given dataset attribute. We will denote by $\mathcal{D}^{\Phi}$ the distribution over training sets $S \mid \Phi(S)$, i.e., those which exhibit property Φ.

We model property inference as a game, similar to how prior work [83, 161] has defined membership inference. Our property inference game is given by Definition 8.2. Essentially, the goal of the adversary is to determine from a model, f, which of a known set of hypothetical dataset properties are exhibited by the training set used to produce f. In our

(a) LFW-12 (b) CIFAR-10

Figure 8.2: Prototypical examples of the watermarks we apply to each dataset.

setting, adversary makes this determination given access to f, as well as the set of possible properties, $\mathcal{P}$, and the ability to train new models on training sets with specific properties in $\mathcal{P}$ (Threat Model 8.2).

Definition 8.2 (Property Inference Game)**.** *Let $\mathcal{P}$ be a collection of k dataset properties, $\Phi_0, \ldots, \Phi_k$. Select j uniformly at random from $[k]$, and draw S according to $\mathcal{D}^{\Phi_j}$; i.e., draw a training set with property Φ_j. Let $f = \mathcal{A}(S)$; i.e., obtain a target model, f, by training on S. Label f according to the boolean vector $\ell = (\forall i \in S \;.\; \Phi_i(S))$.*

The adversary's goal is to predict ℓ (i.e., which properties from $\mathcal{P}$ applied to S) given some auxiliary information, determined by a chosen threat model.

Threat Model 8.2 (Strong White-box PI Adversary)**.** *We assume that the adversary has access the following: (i) the target model function, f, including its parameters, θ (i.e., the adversary has white-box access to f); (ii) the training algorithm, $\mathcal{A}$, used to produce f, including any hyperparameters; (iii) the set of possible properties, $\mathcal{P}$, and (iv) the ability to sample from $\mathcal{D}^{\Phi_i}$ for each property, $\Phi_i \in \mathcal{P}$.*

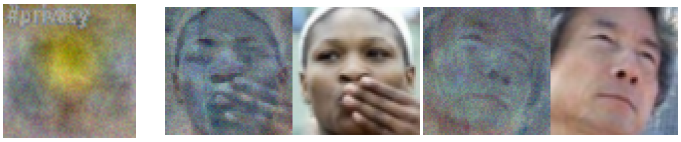
8.1.3 Model Inversion

Model inversion attacks, introduced by Fredrikson et al. [43], aim to reconstruct inputs that are "typical" with respect to a model's training data. Formally, given a model, $f : \mathbb{R}^n \to \mathbb{R}^m$, a given class, $j \in [m]$, and loss function, L, the adversary attempts to find a reconstruction, x', according to Equation 8.1:

$$\underset{x'}{\operatorname{argmin}} \Big\{ L\big(j, f(x')\big) \Big\} \text{ subject to } x' \in [0,1]^n \tag{8.1}$$

In the simplest form of the attack, this is achieved via gradient descent in the input space of the model, starting from a provided seed (e.g., a random initialization). The constraint specifying that x' should be in $[0,1]^n$ is typically used in image domains where pixels are assumed to take values in the $[0,1]$ range. In other domains, alternative boundary constraints may be appropriate.

Typically, model inversion is performed as a white-box attack that relies on computing input gradients of the target model f to optimize Equation 1. For the sake of simplicity, we primarily abide by standard white-box access for the reconstruction attacks in our evaluation (Threat Model 8.3); however, we will also show that fundamentally, the same results can be obtained even *with query access only* (Threat Model 8.4). We draw attention to the fact



(a) (b)

Figure 8.3: (a) An example of a reconstruction obtained from a robustly-trained neural network trained with watermarked data. We see that the watermark, which reads "#privacy," is clearly recognizable despite the fact that the adversary performing the reconstruction had no *a priori* knowledge of its existence. **(b)** An example of explicit instance leakage on a robustly-trained dense network. In each pair of images, a reconstruction is shown on the left, and the most similar training instance (measured by SSIM) is shown on the right for comparison.

that in both threat models, no access to the data, or a distribution like it, is required. This differs from most threat models commonly used in privacy attacks in the literature, which typically require access to auxiliary data. For this reason, we refer to these threat models as "weak" adversaries.

Threat Model 8.3 (Weak White-box Adversary)**.** *We assume that the adversary has access only to the target model, f, including its parameters, θ.*

Threat Model 8.4 (Weak Black-box Adversary)**.** *We assume that the adversary has access only to query the target model, f. That is, the adversary does not have access to the model parameters, θ, but may obtain the logit outputs, $f(x)$, on any input $x \in \mathbb{R}^n$.*

Ultimately, the goal of the adversary is to produce reconstructions in which private properties of the training data are evidently recovered. Qualitatively, this means that a human can learn of their presence and nature upon inspection of the reconstruction. It is in this sense that the reconstruction adversary is "weak," as it does not possess (nor does it rely on) prior knowledge of the nature of the private properties.

While we do not propose any novel model inversion techniques, our work is the first to report on the connection between robustness and the ability for a weak adversary to discover private information encoded by the model through model inversion—even without *a priori* knowledge of the nature or existence of the leaked private properties. Figure 8.3 provides an example of this phenomenon. Figure 8.3a shows a reconstruction of a watermark that was added to the training set, with the express goal of ensuring that the watermark itself is not a predictive feature. Despite this, the watermark is explicitly encoded by the model and leaked through the visualization so that an adversary without *a priori* knowledge of its existence can easily recover it. Figure 8.3b shows reconstructions that resemble individual training instances—capturing aspects of gesture and pose that are unique to those instances, and not necessary for the model's classification task.

8.2 A Bayes-Optimal Membership Inference Attack

We begin by focusing on the problem of membership inference. In this section and the next, we introduce a novel membership inference attack that exploits a model's use of features that are overly-specific to the training set, demonstrating that inappropriate feature-use compromises the privacy of a model. We begin in this section by laying out the core principles

of our attack in an idealized setting where the exact data distribution is known and the model is linear (Section 8.2.1). Under these circumstances, we can more rigorously analyze our attack, justifying it as the Bayes-optimal logistic attack model under our simplifying assumptions. In particular, we show that when specific data-generating assumptions hold, the confidence scores produced by this attack correspond to the true membership probability, and can thus be used for effective, accurate calibration towards high-precision attacks. Using the insights gained from this analysis, we then show how to generalize the attack to settings where the model is linear but the data-generating distribution is unknown, or does not match our previous simplifying assumptions (Section 8.2.2). Finally, we extend the attack to deep models and relate it more clearly to conceptual soundness in Section 8.3.

8.2.1 Membership Inference in an Idealized Setting

Models learn to use features to distinguish between classes, and while some features may be truly discriminative (i.e., function as good predictors on unseen data), others may be discriminative only on the particular training set merely by coincidence. When the model applies features of the latter type to make a prediction, this can be thought of as "evidence" of overfitting regardless of whether the prediction is correct; the salience of a feature coincidental to the training data is suggestive on its own. Similarly, there may be features that are discriminative on the data in general, but not on the training data, which would serve as evidence against training set membership.

To motivate this intuition more formally, we begin by showing how to model an evidence-based attack in an idealized setting where data is distributed according to a known distribution. This provides a simpler illustration of the central ideas used in our real attack, *where we do not make explicit assumptions about the data distribution.* We show that the attack in this setting leads to Bayes-optimal membership predictions on points from that distribution, which suggests that even when the strict assumptions made here are violated, the approach may nonetheless be a strong heuristic.

Generative Assumptions

Recall the setting described in Section 8.1: a model, $\hat{g}$, trained on $S \sim \mathcal{D}^*$, and an adversary that leverages white-box access to $\hat{g}$ to create an attack model, m, that predicts whether a labeled instance, (x, y), belongs to S. We show how membership inference can be analyzed by introducing some assumptions about $\hat{g}$ and $\mathcal{D}^*$.

First we assume that $\mathcal{D}^*$ is given by parameters, μ_y^*, Σ^*, and $p^* = (p_1^*, \ldots, p_C^*)$, such that the labels, y, are distributed according to a Categorical distribution with parameter p^*, and the features, x, are multivariate Gaussians with mean μ_y^* for each label y, and covariance matrix, Σ^* (Equation 8.2).

$$y \sim \text{Categorical}(p^*) \quad x \sim \mathcal{N}(\mu_y^*, \Sigma^*) \tag{8.2}$$

Furthermore, assume that Σ^* is a diagonal matrix, i.e., the distribution of x satisfies the so-called "naive-Bayes assumption," which states that the features are *conditionally independent* given the class label. We will therefore write Σ_{jj}^* as σ_j^{*2}.

Because S is drawn i.i.d. from $\mathcal{D}^*$, in expectation, the class priors, means, and covariance of its features will be given by p^*, μ^*, and Σ^*. However, the *empirical* class priors, means, and covariance will not match these values exactly, as S is a finite sample from $\mathcal{D}^*$. Therefore, let $\hat{p}$ be the empirical class prior of the labels in S, $\hat{\mu}_y$ be the empirical mean of the features

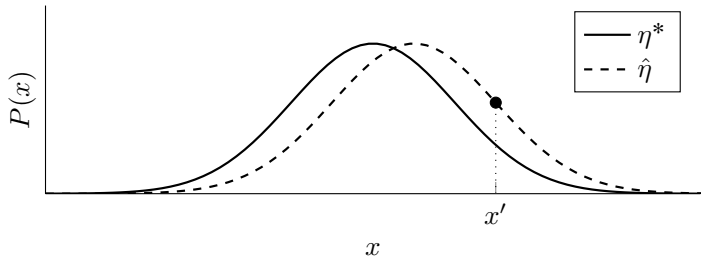


Figure 8.4: Example of two Gaussian distributions, η^* and $\hat{\eta}$. The point x' has a higher probability of being generated by $\hat{\eta}$ than by η^*. Given a prior probability of $1/2$ for being drawn from either distribution, the decision boundary for predicting which distribution a given point was drawn from would be at the intersection of the two curves, and x' would be predicted to have been drawn from $\hat{\eta}$.

in S with class y, and $\hat{\Sigma}$ be the empirical covariance matrix of the features in S. For the purposes of our analysis, we will model the training set as a *distribution*, $\hat{\mathcal{D}}$ parameterized by $\hat{p}$, $\hat{\mu}_y$, and $\hat{\Sigma}$ analogously to $\mathcal{D}^*$. In addition, we will make the corresponding assumption that $\hat{\Sigma}$ is a diagonal matrix.

Under these assumptions, we can now think of the adversary's goal as the following: given a labeled point, (x, y), determine if (x, y) is more likely to have been drawn from $\hat{\mathcal{D}}$ (i.e., $(x, y) \in S$), or $\mathcal{D}^*$. If we momentarily assume that the attacker knows $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, then we can proceed to derive a logistic attack model purely in terms of their respective parameters, namely p^*, $\hat{p}$, μ_y^*, $\hat{\mu}_y$, Σ^*, and $\hat{\Sigma}$—this is, of course, for illustrative purposes, and not realistic for a practical attack; we will dispose of this assumption in Section 8.2.2.

A Simplified Illustrative Example

Let us first ignore the class labels, and consider 1-dimensional features, $x \in \mathbb{R}$. If we have two Gaussian distributions, $\eta^* = \mathcal{N}(\mu^*, \sigma^*)$ and $\hat{\eta} = \mathcal{N}(\hat{\mu}, \hat{\sigma})$, assuming a prior probability of $1/2$ of being drawn from either distribution, x is more likely to have been generated by $\hat{\eta}$ than by η^* when $\mathcal{N}(x \mid \hat{\mu}, \hat{\sigma}) > \mathcal{N}(x \mid \mu^*, \sigma^*)$ (where $\mathcal{N}(x \mid \mu, \sigma)$ is the Gaussian PDF, given by Equation 8.3).

$$\mathcal{N}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{8.3}$$

Thus, we can construct a simple model that predicts whether x was drawn from $\hat{\eta}$ rather than η^* by solving for x in this inequality. When the variances, σ^* and $\hat{\sigma}$, are the same, this produces a linear decision boundary that is a function of $\mu^* - \hat{\mu}$ and σ^*. An example of this is shown pictorially in Figure 8.4.

An Omniscient Attack Model

Our setting is more complicated than the above simple Gaussian example, but as we demonstrate below, the same principle can be applied to mount an attack. Let (X, Y) be random variables drawn at random from either $\hat{\mathcal{D}}$ or $\mathcal{D}^*$, and let t be the probability of drawing from $\hat{\mathcal{D}}$. Let T be the event that (X, Y) was drawn from $\hat{\mathcal{D}}$, which we take to mean that $(X, Y) \in S$.Thus, $\Pr[T] = t$. In keeping with the membership inference definition presented in Section 8.1, we will assume that $t = 1/2$.

We now derive an attack model, m, such that $m^y(x) = \Pr[T \mid X = x, Y = y]$, interpreted to mean that m gives us the probability that a labeled point (x, y) is a member of the training set, S. Because we know t and the parameters of $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, we can derive an estimator for this quantity by applying Bayes' rule and algebraically manipulating the result to fit a logistic function of the log odds. We then make use of the naive-Bayes assumption, allowing us to write the probability of observing x given its label as the product of the probabilities of observing each of x's features independently. The result is linear in the target feature values when $\hat{\sigma} = \sigma^*$, as detailed in Theorem 8.1.

Theorem 8.1. *Let T be a random indicator variable where $Pr[T] = 1/2$. Let $(X, Y) \sim \hat{\mathcal{D}}$ if $T = 1$ and $(X, Y) \sim \mathcal{D}^*$ otherwise, where $\hat{\mathcal{D}}$ and $\mathcal{D}^*$ are given by Equation 8.2 with parameters $(\hat{p}, \hat{\mu}_y, \hat{\Sigma})$ and (p^*, μ_y^*, Σ^*), respectively. Further, assume that $\hat{\Sigma} = \Sigma^*$ is diagonal and $\hat{p} = p^*$. Then $Pr[T \mid X = x, Y = y] = m^y(x)$ where m is given by Equation 8.4, and $\mathcal{s}$ is the sigmoid function given by $\mathcal{s}(x) = (1 + e^x)^{-1}$.*

$$m^y(x) = \mathcal{s}\left(w^{yT}x + b^y\right) \tag{8.4}$$

$$\text{where} \qquad w^y = \frac{\hat{\mu}_y - \mu_y^*}{\sigma^2} \quad b^y = \sum_j \frac{\mu_{yj}^{*2} - \hat{\mu}_{yj}^2}{2\sigma_j^2}$$

Proof. We begin with the expression for $P[T \mid X = x, Y = y]$ and apply Bayes' rule to obtain Equation 8.5.

$$\Pr[T \mid X = x, Y = y] = \frac{\Pr[X = x \mid T, Y = y] \Pr[T]}{\Pr[X = x \mid Y = y]} \tag{8.5}$$

Next, we express Equation 8.5 as a sigmoid function. Since we assume that $\Pr[T] = 1/2$, $\Pr[X = x \mid Y = y]$ can be written according to Equation 8.6 by the law of total probability.

$$\Pr[X = x \mid Y = y] = \frac{1}{2}\left(\Pr[X = x \mid T, Y = y] + \Pr[X = x \mid \neg T, Y = y]\right) \tag{8.6}$$

We then divide by the numerator in Equation 8.5, yielding an expression that can be written as a logistic function (8.7) by noting that for $x > 0$, $\exp(\log\ x) = x$.

$$\begin{aligned}
(8.5) &= \frac{\Pr[X = x \mid T, Y = y]}{\left(\Pr[X = x \mid T, Y = y] + \Pr[X = x \mid \neg T, Y = y]\right)} \\
&= \left(1 + \frac{\Pr[X = x | \neg T, Y = y]}{\Pr[X = x | T, Y = y]}\right)^{-1} \\
&= \left(1 + \exp\left(\log \frac{\Pr[X = x | \neg T, Y = y]}{\Pr[X = x | T, Y = y]}\right)\right)^{-1} \\
&= \mathcal{s}\left(\log \frac{\Pr[X = x \mid T, Y = y]}{\Pr[X = x \mid \neg T, Y = y]}\right)
\end{aligned} \tag{8.7}$$

We notice that $\Pr[X = x \mid T, Y = y]$ is the probability of having drawn x from $\hat{\mathcal{D}}$, given class, y, and similarly, $\Pr[X = x \mid \neg T, Y = y]$ is the probability of having drawn x from $\mathcal{D}^*$, given class, y. Using the Naive-Bayes assumption, i.e., that conditioned on the class, y,

the individual features, x_j, are independent, we obtain Equation 8.8.

$$(8.7) = \mathscr{o}\left(\log \prod_j \frac{\mathcal{N}(x_j \mid \hat{\mu}_{yj}, \hat{\sigma}_j^2)}{\mathcal{N}(x_j \mid \mu^*_{yj}, \sigma^{*2}_j)}\right) \tag{8.8}$$

We then re-write the log of the product as a sum over the log, and observe that the sum can be written as a dot product as in Equation 8.9, which gives the parameters of the Bayes-optimal model for $m^y(x)$.

$$\begin{aligned}(8.8) &= \mathscr{o}\left(\sum_j \frac{(x_j - \mu^*_{yj})^2}{2\sigma^{*2}_j} - \frac{(x_j - \hat{\mu}_{yj})^2}{2\hat{\sigma}_j^2} + \log\left(\frac{\sigma^*_j}{\hat{\sigma}_j}\right)\right) \\ &= \mathscr{o}\left(v^{y\ T} x^2 + w^{y\ T} x + b^y\right) \end{aligned} \tag{8.9}$$

where

$$v^y_j = \frac{1}{2\sigma^{*2}_j} - \frac{1}{2\hat{\sigma}_j^2} \qquad w^y_j = \frac{\hat{\mu}_{yj}}{\hat{\sigma}_j^2} - \frac{\mu^*_{yj}}{\sigma^{*2}_j}$$

$$b^y = \sum_j \left(\frac{\mu^{*2}_{yj}}{2\sigma^{*2}_j} - \frac{\hat{\mu}^2_{yj}}{2\hat{\sigma}_j^2}\right) + \log\left(\frac{\sigma^*_j}{\hat{\sigma}_j}\right)$$

Finally, by assumption the variance is the same in S as in the general distribution, i.e., $\hat{\sigma}_j = \sigma^*_j = \sigma_j$, for all features, j. Thus, v^y from Equation 8.9 becomes zero, so we are left with a linear model for m^y, with weights, w^y, and bias, b^y, given by Equation 8.4. □

An immediate corollary of Theorem 8.1 is that $m^y(x) > 1/2$ is the Bayes-optimal predictor for T, as it predicts the outcome for T that is most likely according to the Bayesian probability given by m.

Notice that the magnitude of the attack model weights given in Theorem 8.1 is large only on features whose means on the training data differ significantly from their means in the distribution, $\mathcal{D}^*$, relative to their variance. This is a manifestation of the intuition described earlier in this section, as the attack model effectively treats those features as its primary "evidence" for deciding membership.

Summary. Features that are more likely in the empirical training distribution, $\hat{\mathcal{D}}$, than in the true "general population" distribution, $\mathcal{D}^*$, serve as evidence for membership. Theorem 8.1 shows how this evidence can be compiled into a linear attack model with parameters w^y and b^y that achieves Bayes-optimality for membership inference when both distributions are known precisely. In Section 8.2.2, we show how to obtain approximate values for w^y and b^y when the distributions $\hat{\mathcal{D}}$ and $\mathcal{D}^*$ are unknown.

8.2.2 Obtaining Attack Parameters from Proxy Models

In practice, it is unrealistic to know the exact parameters defining the distributions $\mathcal{D}^*$ and $\hat{\mathcal{D}}$. In particular, our threat model assumes that the attacker has no *a priori* knowledge of the parameters of $\hat{\mathcal{D}}$ or the elements of S, only that S was drawn from $\mathcal{D}^*$. While we assume white-box access to the target model, $\hat{g}$, we cannot expect that it will explicitly encode $\hat{\mathcal{D}}$;

indeed, $\hat{g}$ is usually parameterized by weights, leaving the distribution parameters underdetermined. Finally, $\mathcal{D}^*$ and $\hat{\mathcal{D}}$ may violate the naive-Bayes assumption, and may moreover be difficult to parameterize directly.

These issues can be largely addressed by observing that the learned weights are sensitive to $\hat{\mathcal{D}}$, and although they may not encode sufficient information to solve for the exact parameters, they can still encode useful information about the differences between $\hat{\mathcal{D}}$ and $\mathcal{D}^*$. To measure these differences, we use a *proxy dataset*, $\tilde{S}$, which is drawn i.i.d. from $\mathcal{D}^*$ (but distinct from S) to train a proxy model, $\tilde{g}$, which is then compared with $\hat{g}$. To control for differences in the learned weights resulting from the learning algorithm, rather than from differences between $\hat{\mathcal{D}}$ and $\mathcal{D}^*$, the proxy model is trained using the same algorithm and hyperparameters as $\hat{g}$ (note that this information is assumed to be known in our threat model). This process can be repeated on many different $\tilde{S}$, using bootstrap sampling when the available data is limited.

For the moment we will continue with the assumption that data is generated according to Equation 8.2. In this part of our analysis we will assume that the target model being attacked is a linear model $\hat{g}(x) = \hat{W}^T x + \hat{b}$, that minimizes 0-1 loss on S for the predictions given by $\text{argmax}_{j\in[m]}\{\hat{g}_j(x)\}$. This is a convex optimization problem that, under our generative assumptions, is minimized when $\hat{W}$ and $\hat{b}$ are given by Equation 8.10.[1]

$$\hat{W}_{jy} = \frac{\hat{\mu}_{yj}}{\hat{\sigma}_j^2} \qquad \hat{b}_y = \sum_j \frac{-\hat{\mu}_{yj}^2}{2\hat{\sigma}_j^2} + \log(\hat{p}) \tag{8.10}$$

Plugging this, and the analogous equation for the proxy model, $\tilde{g}(x) = \tilde{W}^T x + \tilde{b}$, into Equation 8.4 from Theorem 8.1, we see that the weights and biases of the attack model m^y are approximated by $w^y = \hat{W}_{:y} - \tilde{W}_{:y}$ and $b^y = \hat{b}_y - \tilde{b}_y$ respectively, assuming that $\tilde{\mu} = \mu^*$, i.e., that the proxy data reflects the true distribution mean. This is summarized in Observation 8.2, which leads to a natural attack as shown in Algorithm 8.1. We call this the bayesian-wb attack.

Observation 8.2. *For linear softmax model, $\hat{g}$, with weights, $\hat{W}$, and biases, $\hat{b}$; and proxy model, $\tilde{g}$, with with weights, $\tilde{W}$, and biases, $\tilde{b}$, the Bayes-optimal membership inference model, m, on data satisfying Eq. 8.2 is approximated by Equation 8.11 when $\hat{g}$ and $\tilde{g}$ have converged on their respective training sets and when $\tilde{\mathcal{D}}$ is a good approximation for $\mathcal{D}^*$.*

$$m^y = \sigma\left(w^{y\,T} x + b^y\right) \tag{8.11}$$

$$\text{where} \qquad w^y = \hat{W}_{:y} - \tilde{W}_{:y} \quad b^y = \hat{b}_y - \tilde{b}_y$$

Notice that Observation 8.2 gives the weights and biases of m^y in terms of only the observable parameters of the target and proxy models. This is therefore possible *even when the distributions, $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, are unknown.* Furthermore, while Observation 8.2 is derived and stated using relatively strong generative assumptions, we find that this attack is nevertheless often effective when these assumptions do not hold.

Disposing of Generative Assumptions

One way of viewing the bayesian-wb attack is that it weights membership predictions by measuring a sort of displacement between the weights of the target model and the ideal

[1] See [105, Slide 20] for details.

Algorithm 8.1: The Linear bayesian-wb MI Attack

def `createAttackModel` *($\hat{g}$, $\tilde{S}$)*:
  $\tilde{g} \leftarrow$ `trainProxy`$(\tilde{S})$
  $w^y \leftarrow \hat{g}.W_{:y} - \tilde{g}.W_{:y} \qquad \forall y \in [m]$
  $b^y \leftarrow \hat{g}.b_{:y} - \tilde{g}.b_{:y} \qquad \forall y \in [m]$
  return $\lambda(x, y) : \sigma\big(w^{yT}x + b^y\big)$
def `predictMembership` *(M, x, y)*:
  return 1 if $M^y(x) > \frac{1}{2}$ else 0

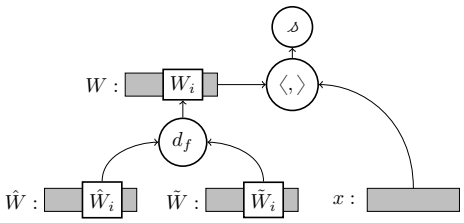


Figure 8.5: Illustration of the generalized attack model. A learned displacement function, d, is applied element-wise to the weights of the target and proxy model to produce attack model weights, W. The inner product of W and x is then used to make the membership prediction. *Not pictured: d is also applied to the biases, $\hat{b}$ and $\tilde{b}$, to produce b, which is added to the result of the inner product.*

weights of the true distribution as approximated by the proxy model. Let D be a *displacement function* that is applied element-wise to the weights of the model—that is for some $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, on vectors $v, u \in \mathbb{R}^n$, let $D(x, y) = (d_f(v_1, u_1), \ldots, d(v_n, u_n))$. We can express the bayesian-wb attack via a such a displacement function, namely, $w^y = D(\hat{W}_{:y}, \tilde{W}_{:y})$ and $b^y = D(\hat{b}_y, \tilde{b}_y)$, by letting $d(v_i, u_i) = v_i - u_i$, i.e., by letting D be element-wise subtraction.

As per Observation 8.2, element-wise subtraction is optimal for membership inference under the Gaussian naive-Bayes assumption, but it may be that for other distributions, a different displacement function is more appropriate. More generally, we can represent the displacement function as a neural network, and train it using whatever data is at hand.

Figure 8.5 illustrates this approach, which we call the nn-wb attack. A learned displacement function, d, is applied element-wise to $\hat{W}$ and $\tilde{W}$ to produce attack model weights, W, and to $\hat{b}$ and $\tilde{b}$ to produce attack model biases, b. It then predicts the probability of membership as $\sigma\big(W_{:y}^T x + b_y\big)$.

As d is applied element-wise to pairs of weights, we model D as a 1-dimensional convolutional neural network, where the initial layer has a kernel size and strides of 2 (i.e., the kernel is applied to one element of $\hat{W}_{:y}$ and one element of $\tilde{W}_{:y}$), and subsequent layers have a kernel size and stride of 1.

In order to learn the weights of D, we partition $\tilde{S}$ into an "in" dataset, $\tilde{S}^1$, and an "out" dataset, $\tilde{S}^0$. We train a shadow target model, $\check{g}$, on $\tilde{S}^1$ and a proxy model, $\tilde{g}$, on $\tilde{S}^0$. We then create a labeled dataset, T, where the features are the weights and biases of $\check{g}$, the weights and biases of $\tilde{g}$, and x; and the labels are 1 for x belonging to $\tilde{S}^1$ and 0 for x belonging to $\tilde{S}^0$. Finally we train to find the parameters to D that minimize the 0-1 loss, $\mathcal{L}$, of the nn-wb attack on T. We can increase the size of T to improve the generalization of the attack by repeating over multiple in/out splits of $\tilde{S}$. This procedure is described in Algorithm 8.2.

Algorithm 8.2: The Linear nn-wb MI Attack

def `createAttackModel` *($\hat{g}$, $\tilde{S}$, N)*:
  for $i \in [N]$ **do**
    $\tilde{S}_i^1, \tilde{S}_i^0 \leftarrow \texttt{split}_i(\tilde{S})$
    $\check{g}_i \leftarrow \texttt{trainShadow}(\tilde{S}_i^1)$
    $\tilde{g}_i \leftarrow \texttt{trainProxy}(\tilde{S}_i^0)$
  $T \leftarrow [(\check{g}_i.W_{:y},\ \tilde{g}_i.W_{:y},\ \check{g}_i.b_y,\ \tilde{g}_i.b_y,\ x,\ \ell)]$
    $\forall (x,y') \in \tilde{S}_i^\ell : y' = y,\quad \forall y \in [C],\quad \forall \ell \in \{0,1\},\quad \forall i \in [N]$
  $D \leftarrow \underset{D'}{\text{argmin}} \left\{ \underset{(\hat{w},\tilde{w},\hat{b},\tilde{b},x,\ell)\in T}{\mathbb{E}} \left[\mathcal{L}\big(\mathcal{s}\big(D'(\hat{w},\tilde{w})^T x + D'(\hat{b},\tilde{b})\big), \ell\big) \right] \right\}$
  $\tilde{g} \leftarrow \texttt{trainProxy}(\tilde{S})$
  return $\lambda(x,y) : \mathcal{s}\big(D(\hat{g}.W_{:y}, \tilde{g}.W_{:y})^T x + D(\hat{g}.b_y, \tilde{g}.b_y)\big)$
def `predictMembership` *(m, x, y)*:
  return 1 if $m^y(x) > \frac{1}{2}$ else 0

	omniscient	bayesian-wb	nn-wb *(min capacity)*	nn-wb *(extra capacity)*
$n = 100$	0.618	0.605	0.602	0.590
$n = 200$	0.577	0.570	0.563	0.562
$n = 400$	0.568	0.550	0.547	0.542

Table 8.6: Comparison of the bayesian-wb and nn-wb attacks to an *omniscient* attack, which has knowledge of $\hat{\mu}$, μ^*, and σ, and thus can use Theorem 8.1 directly without the use of a proxy model. In one case, the nn-wb attack was given the minimum capacity to reproduce the bayesian-wb attack, i.e., d is simply a weighted sum of $\hat{W}_i$ and $\tilde{W}_i$. In another case, the nn-wb attack was given excess capacity, with 16 hidden units in d. Three target models, trained on synthetic Gaussian naive-Bayes data with training set sizes of 100, 200, and 400, were attacked.

8.2.3 Efficacy of Proxy Models

In Section 8.2.1, we derived the Bayes-optimal membership inference attack on Gaussian data satisfying the naive-Bayes condition. The weights of the optimal membership predictor for this case, given by Theorem 8.1, are a function of the empirical training distribution parameters and true distribution of the data, which, of course, would be unknown to an attacker. Section 8.2.2 describes how to address this, using a *proxy model* to capture the difference between the data used to train the target model and the general population.

To measure the extent to which this proxy model approach can recover the Bayes-optimal attack, we created a synthetic dataset matching the generative assumptions used for our derivation. The synthetic data were generated with 10 classes, 75 features, and 400, 800, or 1,600, records, with an equal number of records per class. The features, x_j, of the synthetic data were drawn randomly from a multivariate Gaussian distribution with parameters, μ_y (for each class, y) and Σ, where μ_{yj} was drawn uniformly at random from $[0, 1]$, and Σ was a diagonal matrix with Σ_{jj} drawn uniformly at random from $[0.5, 1.5]$. Each dataset was divided into three groups, with 1/4 of the records used to train a linear target model (these comprise the "members"), 1/4 used to test the attack performance on non-members, and 1/2 used as hold-out auxiliary data for the attacker to use to construct the attack model (i.e., to train proxy models, etc.).

Table 8.6 demonstrates the effectiveness of the proxy model in our attack, by comparing our bayesian-wb attack using a proxy model to an "omniscient" attack, which uses Equation 8.10 directly, with knowledge of the empirical means and covariances of the set of members and the set of non-members. We can consider the omniscient attack as giving an upper bound on the expected accuracy of a white-box attack on Gaussian naive-Bayes data, as it is the true Bayes-optimal attack (while bayesian-wb is the approximate Bayes-optimal attack according to Proposition 8.2). Our bayesian-wb attack achieves on average 84% of the advantage of the omniscient attack, suggesting that the proxy model was able to approximately capture the general distribution as necessary for the purpose of detecting the target model's idiosyncratic use of features.

We also further generalized the bayesian-wb attack to use a learned displacement function that may be more appropriate for distributions that don't resemble the Gaussian naive-Bayes assumption (Algorithm 8.2). While we find that this nn-wb attack often generalizes to arbitrary distributions better than the bayesian-wb attack, because its displacement function is learned, it is possible for the nn-wb attack to overfit.

Table 8.6 also shows the accuracy of the nn-wb attack on Gaussian naive-Bayes data. When the neural network representing the displacement function is given exactly enough capacity to reproduce the bayesian-wb attack, nn-wb recovers on average 94% of the advantage of the bayesian-wb attack. Upon inspecting the weights of the displacement network, we find that nn-wb learns almost exactly element-wise subtraction, demonstrating its potential to learn the optimal displacement function. When given excess capacity, the nn-wb attack performs only marginally worse, achieving on average 92% of the minimal nn-wb attack's advantage (86% of bayesian-wb), suggesting that nn-wb is not highly prone to overfitting.

8.3 Exploiting Idiosyncratic Feature-use in DNNs

We have now seen how to approximate the Bayes-optimal estimator for membership prediction using the weights of a linear target and proxy model; in this section, we extend the same reasoning to deep models. However, as deep networks learn novel intermediate representations, the *semantic meaning* of an internal feature at a given index—i.e., the data characteristic that it associates with—will not necessarily line up with the semantic meaning of the corresponding internal feature in another model [9, 163]. This holds even when the models share identical architectures, training data, and hyper-parameters, as long as the randomization in the gradient descent is distinct between the models. In general, the only features for which two models will necessarily agree are the models' inputs and outputs, as these are not defined by the training process.

This poses a challenge for any white-box attack that attempts to extend the "shadow model" approach [128] developed for black-box membership inference. Consider such an approach, which learns properties of internal features that indicate membership—involving activations of specific neurons, gradients, or any other quantity—from shadow models. Any such property must make reference to specific internal features within the shadow model, but even if the target model contains internal features that match these properties, they are unlikely to reside at exactly the same location within the network as they do in the shadow model. *This is why previous white-box attacks [107] require large amounts of the target model's training data*; rather than learning attack models from shadow models, they are forced to learn them from the target model itself and its training data. Such methods thus learn differences between the training and auxiliary data, but do not offer any general

insights as to how those differences manifest in the model.

To circumvent this limitation, one must either construct a mapping between internal features in the shadow and target models, or fix the feature representation in the shadow model to preserve semantic meaning between the two. In this section, we show how to accomplish the latter by constructing a series of *local linear approximations* of the network (Section 8.3.1), one for each internal layer, that operate on the feature representation of the target model. Because each approximation is linear, we can apply any of the attacks from Section 8.2 to each approximation, and combine the results (Section 8.3.2) to form an attack model for the full network.

8.3.1 Local Linear Approximations of Deep Models

We define a local linear approximation in terms of a *slice*, $(g, h($, which decomposes a deep network, f, into two functions, g and h, such that $f = g \circ h$. Intuitively, a slice corresponds to a layer of the network, where h yields the internal features computed by the layer, and g computes the output of the model from these features.

For the slice at the penultimate layer of the network, g is simply a linear model acting on features computed by the rest of the model. In this case, no local approximation is needed and the bayesian-wb (Algorithm 8.1) and nn-wb (Algorithm 8.2) attacks can be applied directly to g using internal features that are computed by h.

For slices lower in the network, g is no longer linear, but we can approximate the way in which g makes use of its features at a particular point by constructing a linear model that agrees with it *at that point* and faithful to its local behavior. To do this, we make use of an *influence measure* (see Chapter 3) over the inputs of g to its computed output for each point. Recall that given a model, f, a point, x, and feature, j, the *influence* of a feature x_j on f is a quantitative measure of x_j's potential impact the output of f. Essentially, our goal is to replace the model weights in Algorithms 8.1 and 8.2 with the corresponding feature *influences*, which are the natural analogues of the feature weights in linear models.

We can think of this attack as comparing the explanations between the target and proxy models to find evidence of idiosyncratic feature use that might be linked to the training set. For instance, in our example in Figure 8.1 (Section 8.1), if influence-guided explanations reveal that the target model uses a "pink background" feature to identify Tony Blair, while the proxy model does not, the resulting attack model will view pink backgrounds as evidence of membership in images of Tony Blair.

We now specify the parameters we will use for our influence measure when mounting our attack. Recall that the preprocessing axiom of Internal Influence (Axiom 4.4) states that the Internal Influence of a slice, $\chi(g \circ h, q, \mathcal{D}_{\mathcal{X}})$ is equivalent to the internal influence on g using the distribution $\mathcal{D}_{\mathcal{Z}}$, derived by raising $\mathcal{D}_{\mathcal{X}}$ through h. For our purposes, we will specify $\mathcal{D}_{\mathcal{Z}}$ directly. Specifically, when predicting on a point, x, we let $\mathcal{D}_{\mathcal{Z}}^{x}$ be a uniform distribution over the line segment connecting the zero-vector to $h(x)$; recall that this is the distribution implicitly used by Integrated Gradients. This results in a measure satisfying the *efficiency* property (see Chapter 3), stated for our context in Equation 8.12.

$$\sum_{j} \chi_j^{\mathsf{input}}(g, q, \mathcal{D}_{\mathcal{Z}}^{x}) \cdot (z_j - 0) = q \circ g(z) - q \circ g(0) \tag{8.12}$$

Thus, we see that for $z = h(x)$, we have that $q \circ g(z) = \chi^{\mathsf{input}}(g, q, \mathcal{D}_{\mathcal{Z}}^{x})^T z + g(0)$, giving us a linear approximation of g at x as desired. Since χ^{input} satisfies linear agreement (Axiom 4.1), this linear approximation is exact when g is a linear function. Using this approximation,

Algorithm 8.3: The Deep bayesian-wb MI Attack

def `createAttackModel` *($\hat{g} \circ \hat{h}$, $\tilde{S}$)*:
  $\tilde{S}' \leftarrow [(\hat{h}(x), y)$ for $(x, y) \in \tilde{S}]$
  $\tilde{g} \leftarrow$ `trainProxy`$(\tilde{S}')$
  $w^y \leftarrow \lambda(z) : \chi(\hat{g} \circ \hat{h}, P_0^z)_y - \chi(\tilde{g} \circ \hat{h}, P_0^z)_y \quad \forall y \in [C]$
  $b^y \leftarrow \hat{g}(0)_y - \tilde{g}(0)_y \quad \forall y \in [C]$
  return $\lambda(x, y) : \mathcal{s}\left(w^y (\hat{h}(x))^T \hat{h}(x) + b^y\right)$
def `predictMembership` *(m, x, y)*:
  return 1 if $m^y(x) > \frac{1}{2}$ else 0

we can apply the bayesian-wb and nn-wb attacks to an arbitrary slice of a deep network, by replacing g with its linear approximation. Note that this gives a separate set of weights for each input, x (hence why we call the approximation "local"); however, our attacks are parametric in the weights of the target model, so only a single attack model is necessary. The modification of Algorithm 8.1 for an arbitrary slice of a target deep network is detailed in Algorithm 8.3. An analogous modification of Algorithm 8.2 follows as well, by simply replacing each reference to weights with influence measurements, but is omitted for the sake of brevity.

As an aside, we note that the way we presented is not the only way of obtaining a linear approximation of the network using influence. In general, any distribution of interest could be chosen, as long as the intercept of the linear approximation is adjusted to make the approximation agree with the model. For example, using a single-point distribution of interest is also natural, as it yields the tangent plane to the network at the point under consideration. The exploration of alternative distributions of interest for mounting this attack is left as future work.

Summary. We can generalize the attacks given by Algorithms 8.1 and 8.2 to apply to an arbitrary layer of a deep target network by replacing the weights with their natural generalization, *influence*. This allows us to perform membership inference by detecting anomalies in the way in which a target model uses its learned features. Because influence allows us to create a faithful local linear approximation of the model for any given point, this generalized attack follows from the same analysis on linear models from Section 8.2. In Section 8.3.2, we suggest a method for combining attacks targeting each individual layer to create an attack that utilizes white-box information from *all* the layers of a deep network.

8.3.2 Combining Layers

The results of Section 8.3.1 allow us to leverage overfitting in each layer learned representations employed by the target model towards membership inference. Attacks on different layers may pick up on different signals, but because the model's internal representations are not independent across layers, we cannot simply concatenate the approximated weights of each layer and treat it as an attack on a single model. Instead, we make use of a *meta model*, which learns how to combine the outputs of the individual layer-wise attacks. The meta model takes the confidences of the attack defined in Section 8.3.1 applied to each layer, and outputs a single decision.

To train a meta model, m', to attack target model, f, we partition $\tilde{S}$ into two parts, $\tilde{S}^1$ and $\tilde{S}^0$. We train a shadow target model, $\check{f}$, on $\tilde{S}^1$. Then, for each layer, i, in f, we train an attack model, m_i, on the i^{th} layer of $\check{f}$, as described in Section 8.3.1. We then construct a training set, $T = T^1 \cup T^0$, such that $(x', y') \in T^1$ is constructed as $(x'_i, y') = (m_i^y(x), 1)$ for $(x, y) \in \tilde{S}^1$, and $(x', y') \in T^0$ is constructed as $(x'_i, y') = (m_i^y(x), 0)$ for $(x, y) \in \tilde{S}^0$. We can increase the size of T by creating multiple random partitions of $\tilde{S}$. Finally, we train m' on T.

When building a meta model for the nn-wb attack, we can train m' jointly with the displacement metric, d, rather than first learning a nn-wb attack on each layer. We also use a separate distance metric, d_i for each layer, i, of f.

8.3.3 Evaluation

We now present the results First and foremost, these results demonstrate that a model's use of features may compromise the privacy of its training data. Essentially, both variants of our attack operate on faithful explanations of the model's behavior. The success of the attack depends on the extent to which the explanations uncover idiosyncratic feature-use that identifies specific training instances. To the extent that this is generally inappropriate, we view this as a violation of conceptual soundness—the ideal is for a model to learn general methods for prediction, not for it to memorize one-off patterns.

Experimental Setup

We begin by presenting details on the attack methods, metrics, datasets, target models, and methodology used in our experiments.

Attack Methods. We evaluate our bayesian-wb and nn-wb attacks in comparison to two other baseline approaches which both use only black-box access. By far the simplest membership inference attack, which we dub the "naive" attack (referred to as naive in our evaluation), follows from the fact that generalization error necessarily leads to membership vulnerability [161]. Given a data point and its true label, the attacker runs the model and observes whether its predicted label is correct. If it is, then the attacker concludes that the point was in the training set; otherwise, the point is presumed a non-member. This attack serves as a simple baseline that is more informative than the 50% baseline provided by an adversary that guesses randomly. We also consider the black-box shadow model attack [128], which was briefly introduced in Section 8.1. We refer to this attack as shadow-bb in our evaluation. This serves as a baseline for state-of-the-art membership inference.

Metrics. We compare the performance of the attacks in our evaluation on both accuracy and precision. Because an adversary that guesses randomly achieves 50% accuracy, we will often opt to describe the *advantage* of an attack [161], given by $2(a - 0.5)$ where a is the accuracy. Essentially, advantage scales accuracy to the 50% baseline to yield a measure between -1 and 1. While advantage is an indicator of the degree to which private information is leaked by the model, it does not necessarily capture the severity of the threat posed to any given individual in the training set. From this perspective, a privacy violation occurs if *any* of the points can be confidently identified by the adversary—this is arguably a greater threat than if the adversary were to identify every training member with very low confidence. Thus, we also consider *precision*—the fraction of predicted members that were true members of

model	*# row*	*# feat.*	*# class*	*train acc.*	*test acc.*
Breast Cancer (BCW)	569	30	2	0.987	0.944
Pima Diabetes (PD)	768	8	2	0.789	0.756
Hepatitis (Hep)	155	19	2	0.997	0.810
German Credit (GC)	1000	20	2	0.937	0.701
Adult	48841	99	2	0.861	0.849
MNIST	70k	784	10	0.998	0.987
LFW	1140	1850	5	0.993	0.829
CIFAR-10	60k	3072	10	0.996	0.664
CIFAR-100	60k	3072	100	0.977	0.312

Table 8.7: Characteristics of the datasets and models used in our experiments.

the training set—as a key desideratum for the attacker. In order for an attacker to reach confident inferences, precision must be appreciably greater than $1/2$. If no points are predicted to be members, we define precision to be $1/2$.

Datasets. We performed experiments over nine publicly available classification datasets derived from real-world data. Among the classification datasets were *Adult*, *Pima Diabetes* (obtained from the UCI Machine Learning Repository); *Breast Cancer Wisconsin*, *Hepatitis*, *German Credit*, *Labeled Faces in the Wild* (obtained from scikit-learn's `datasets` API); *MNIST* [79], *CIFAR-10*, and *CIFAR-100* [76]. Table 8.7 shows the characteristics of each of these datasets.

Target Models. The target models we used to conduct our experiments include both simple dense networks, and convolutional neural networks. Each model was trained until convergence with categorical cross-entropy loss, using SGD with a learning rate of 0.1, a decay rate of 10^{-4}, and Nesterov momentum. For non-image real data, we used a dense network architecture with one hidden layer and *ReLU* activations. For datasets with n features, we employed $2n$ hidden units, followed by a softmax layer with one unit per class. For image data, we used a CNN architecture based on LeNet, with two convolutional layers with 5×5 filters and 20 and 50 output channels respectively (each convolutional layer is followed by a max pooling layer), followed by a fully connected layer with 500 neurons. We trained CNNs with a 25% dropout rate following each pooling layer, and a 50% dropout rate following the fully connected layer. Each target model consists of a pair containing an architecture and a dataset; we will refer to each model by its dataset abbreviation given in Table 8.7. The train and test accuracy for each of the target models used in our evaluation are given in the final two columns of Table 8.7.

Methodology. When evaluating each attack, we randomly split the data into three disjoint groups: *train*, *test*, and *hold-out*. The train and test groups were each comprised of one fourth of the total number of instances, and the hold-out group contained the remaining one half of the instances. The target model was trained on the train group, while the attacks were allowed to make use of the hold-out group only. The attack model's predictions were evaluated on the train group (members) and the test group (non-members). Each experiment was repeated 10 times over different random samplings of the data split, and the results were averaged.

	accuracy				*precision*			
model	naive	shadow-bb	bayesian-wb	nn-wb	naive	shadow-bb	bayesian-wb	nn-wb
BCW	0.522	0.500	0.514	**0.523**	0.511	0.500	**0.545**	0.528
PD	0.517	0.508	0.517	**0.519**	0.511	0.515	0.537	**0.561**
Hep	0.595	0.553	0.605	**0.618**	0.552	0.528	0.562	**0.609**
GC	0.618	0.582	**0.623**	0.622	0.572	0.547	0.603	**0.637**
Adult	0.506	**0.524**	0.507	0.516	0.504	0.514	0.512	**0.516**
MNIST	0.506	0.506	**0.575**	0.521	0.503	0.506	0.578	**0.640**
LFW	0.582	0.597	0.618	**0.619**	0.545	0.557	0.581	**0.586**
CIFAR-10	0.666	0.684	0.686	**0.709**	0.600	0.605	0.638	**0.646**
CIFAR-100	0.831	0.847	0.847	**0.872**	0.757	0.766	0.770	**0.792**

Table 8.8: Comparison of the accuracy and precision of bayesian-wb and nn-wb with naive and shadow-bb attacks. Best results for each dataset are shown in bold.

For the bayesian-wb attack, we trained 10 proxy models on random samples from the hold-out group, and took the mean of their approximated weights at each point for added stability of results. When attacking dense networks, we performed the attack on the final layer of the network using Algorithm 8.1. When attacking LeNet models, we used a meta attack model targeting each of the five internal layers (described in Section 8.3.2) that was trained on data from 10 shadow models trained on 10 bootstrapped samples from the hold-out group. We used a small dense network with 16 hidden neurons for the meta model and trained it for 32 epochs with the Adam optimizer [75].

For the nn-wb attack, we construct an attack model that learns a displacement function, D_i, for each layer, i, of the network, and combines the results with a meta attack model, M. The attack model was trained for 32 epochs with Adam, using data from 10 shadow models trained on bootstrapped samples from the hold-out group. As suggested in Section 8.2.2, we modeled each D_i as a convolutional neural network. In each experiment, the networks modeling M and each D_i had at most one hidden layer, with n_M and n_D hidden units, respectively (in our experiments each D_i used the same architecture, though this need not be the case in general). In order to determine n_M and n_D for each dataset, we created a validation set using 10 shadow models trained on different random splits of the hold-out group, and performed a parameter sweep over n_M, n_D. We then took the n_M and n_D yielding the highest validation accuracy for each target model. We find that because the attack model is highly regularized via its restrictive architecture, the validation accuracy is a reasonably good indicator of the test accuracy, making it a useful tool for hyper-parameter tuning.

In each experiment, the shadow-bb attack was trained using 10 shadow models trained on 10 samples from the hold-out group (see [128] for more details on this attack).

Evaluation Results

We now compare our approach to previous work, namely, shadow-bb [128]. In particular, we compare *(1)* performance in terms of accuracy, precision, and recall; and *(2)* the reliability of the attack confidence when used to calibrate for higher precision. In short, our results show that both bayesian-wb and nn-wb outperform shadow-bb, and can be more reliably calibrated to achieve confident inferences for the attacker. Furthermore, even on some well-generalized models, on which shadow-bb and naive fare poorly, our attacks can be calibrated to make confident inferences, and sometimes also achieve non-trivial advantage. Finally,

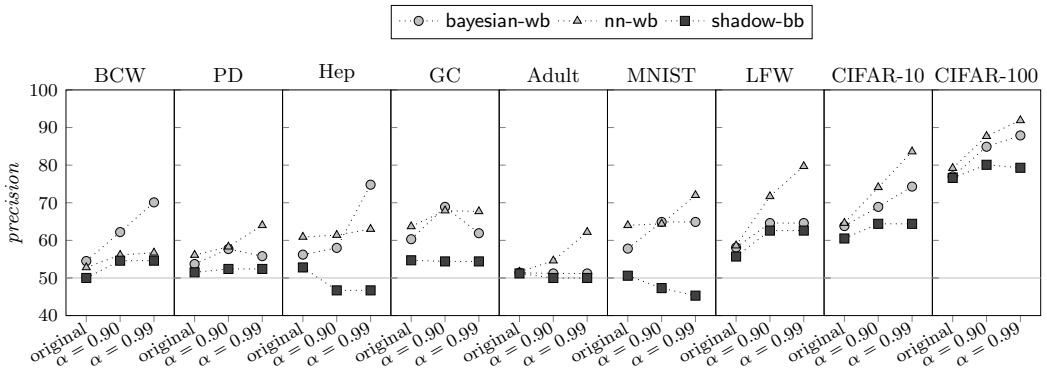


Figure 8.9: Precision of the bayesian-wb, nn-wb, and shadow-bb attacks, calibrated by setting the decision threshold to the α^{th}-percentile prediction confidence (with $\alpha = 0.90$ and $\alpha = 0.99$), compared to the precision with no calibration (i.e., using $1/2$ as the decision threshold for predicting members). An upward slope in the plot indicates a greater degree of calibration.

we find that there is often little advantage to shadow-bb over naive, both because shadow-bb often performs comparably to naive, and because shadow-bb does not always produce calibrated confidence scores.

Performance. Table 8.8 shows the accuracy and precision of naive, bayesian-wb, nn-wb, and shadow-bb. We see that both bayesian-wb and nn-wb are consistently more accurate and precise than naive and shadow-bb. At least one of bayesian-wb or nn-wb obtains the highest accuracy of the four methods on each target model except the one trained on the Adult dataset, and *both* outperform the other two methods in terms of precision in all cases. In some cases, the improvement in accuracy of at least one of our attacks over prior work is by as much as seven percentage points, though in others our accuracy is only modestly better; however, in terms of precision, the difference is more pronounced in almost every case (typically greater by at least five percentage points).

Our results for the performance of shadow-bb are roughly in line with previously reported results for shadow-bb on the datasets which have been used for evaluation in prior work (Adult, MNIST, LFW, CIFAR-10, and CIFAR-100) [121, 128]. On CIFAR-10 and CIFAR-100, our results are slightly lower than the results reported for shadow-bb by Shokri et al. [128], however, our target models trained on CIFAR-10 and CIFAR-100 use dropout and have a lower generalization error than the models in the attacks reported by Shokri et al., which most likely accounts for this small discrepancy.

Calibration. As argued previously, one of the key desiderata of a membership inference attack is precision. In order to calibrate an attack for precision, the confidence output by the attack must be informative. Here, we examine the calibration of the confidence outputs of our attacks compared to shadow-bb (naive does not provide a confidence score with which to calibrate). We find that increasing the decision threshold of the bayesian-wb and nn-wb attacks has a positive effect on precision. In particular, we can choose to predict "member" only on points that receive the highest confidence scores among points from the hold-out set. As the hold-out points are presumed to be non-members, setting the confidence threshold to, e.g., the 99^{th}-percentile confidence score should admit a false positive rate of approximately

1%. If the true members are more likely to have high confidence scores, increasing the decision threshold in this way will improve the precision of the attack.

We see that this is indeed the case for the bayesian-wb and nn-wb attacks. Figure 8.9 shows the precision of bayesian-wb, nn-wb, and shadow-bb as the decision threshold is raised from 1/2 to the 90^{th} and 99^{th} percentile confidence score measured on the hold-out set. For bayesian-wb and nn-wb, this results in an increase in precision in all cases, often by 10 or more percentage points. On all convolutional models, nn-wb is able to be calibrated to upwards of 75% precision. Notably, this includes the model trained on MNIST, which has only 1.1% generalization error. This implies that *privacy violations are a threat even to well-generalized models*, since our attack is able to confidently (with at least 75% confidence) identify a subset of training set members. On the dense models, the calibration is slightly less consistent; however, here bayesian-wb is able to obtain over 70% precision on the models trained on the Breast Cancer Wisconsin and Hepatitis datasets.

While Figure 8.9 demonstrates that applying our calibration heuristic to bayesian-wb and nn-wb consistently increases the precision, we see that this is not always the case for shadow-bb. In some cases, the precision of shadow-bb is *decreased* by increasing the decision threshold. In fact, occasionally, the average confidence on non-members is higher than that of members, leading to a precision slightly less than 50%. This may be a result of the shadow model overfitting to the hold-out data. When we are able to increase the precision of shadow-bb using its confidence output, the gains are less impressive, suggesting the probability outputs of shadow-bb are less well-calibrated.

These results indicate that our attacks can accurately determine which points are most likely to be members, based on the specific observed behavior on those points. By contrast, the naive approach works based on a general heuristic that does not distinguish between characteristics of specific points. The same appears to be true of shadow-bb.

Performance on Well Generalized Models. While some of the models we used to evaluate our attacks had a generalization error of 10% or more, we also evaluated on several datasets for which the learned model was far less overfit, including MNIST (1.1% generalization error), Adult (1.2%), Pima Diabetes (3.4%), and Breast Cancer Wisconsin (4.3%). While on PD and BCW, our attacks only slightly outperform naive, on MNIST and Adult, our attacks do substantially better: on the model trained on Adult, nn-wb achieves an advantage 2.6 times greater than the advantage achieved by naive. Even more impressively, on MNIST, nn-wb and bayesian-wb achieves an advantage 3.5 and 12.5 times greater than the advantage achieved by naive, respectively. On the other hand, shadow-bb fares poorly on all of these datasets except for Adult, typically achieving less than 2% advantage, suggesting that our attacks are better at picking up overfitting that is not manifested as downstream errors in the model's predictions on unseen points.

In addition to the cases where our attacks achieve relatively high advantage against well-generalized models, we find that when calibrated, our attacks achieve as high as 75% precision on MNIST, and 70% precision on Breast Cancer Wisconsin, again underscoring the threat of privacy violations for well-generalized models. While it is clear that a greater degree of overfitting (in the traditional sense of a discrepancy between the train and test errors) makes it easier for an adversary to mount *any* attack, the relative success of our attacks over naive on well generalized models suggests that the white-box information is useful even when the model does not leak information through incorrect predictions on the test set.

Similarity of shadow-bb and naive Results. Table 8.8 reveals that often, shadow-bb has performance comparable or even worse than naive, particularly on well-generalized target models. This is likely a product of the attack model overfitting to idiosyncrasies in the shadow model's output that are unrelated to the target model. On deep models with significant overfitting, shadow-bb performs slightly better than naive, however, we found that its behavior was not significantly different from that of naive; for example, on LFW, naive recovers 88% of the exact correct predictions made by shadow-bb. This supports the intuition that the features used by the shadow model approach (i.e., the softmax outputs) are not fundamentally more well-suited to membership inference than those used by the naive method (i.e., the correctness of the predictions). This is perhaps unsurprising, as the softmax outputs are likely to coincide largely with the correctness of the prediction—for correct predictions, the softmax will likely have high confidence on the correct class, regardless of whether the point was a member or not; and similarly for incorrect predictions, the softmax will likely have more entropy.

Conclusions. The fact that our attacks outperform prior state-of-the-art membership inference attacks demonstrates the value of our insights regarding the mechanisms underlying overfitting. Our attack is designed to solely leverage the way in which the target model uses features in comparison to a baseline, which we find accounts for as much membership leakage as has been extracted by any attack thus far, solidifying the significance of the analysis in this chapter for our understanding of overfitting and its relation to conceptual soundness.

8.4 Model Inversion in Robust Models

We now shift our focus from membership inference to attacks that aim to directly reconstruct private information from the training data that may be encoded by a model. The use of private information can be considered conceptually unsound, particularly when such behavior is not entailed by the model's learning objective. This is especially worrisome if the model encodes private information in a manner that is easily recoverable by a weak adversary (i.e., one provisioned with little auxiliary information about the private information it seeks). As we will see, this concern is related to robustness. Specifically, in line with our observations at the conclusion of Part II, although robust models address certain sources of conceptual unsoundness, the danger of unsound encoding of private features is amplified in robust models. This suggests that work to protect the privacy of robust models is of special importance.

Recent work has suggested that privacy vulnerabilities and robustness to adversarial examples may be related concerns. For example, robustly-trained models have been found to be more vulnerable to membership inference attacks than their non-robust counterparts [136, 162]. The nature of these findings primarily implicates the observation that robustly-trained models generalize poorly on their robust objective; that is, previously unseen points are far more susceptible to adversarial perturbations than points seen during training. These findings are also in line with prior work on membership inference on standard models, which has drawn a strong connection between generalization error and susceptibility to common types of membership inference attacks [83, 116, 161].

Our work contributes new evidence that *another mechanism is also at play regarding the interaction between privacy and robustness*, namely, the types of features encoded by a model and *how they are encoded*.

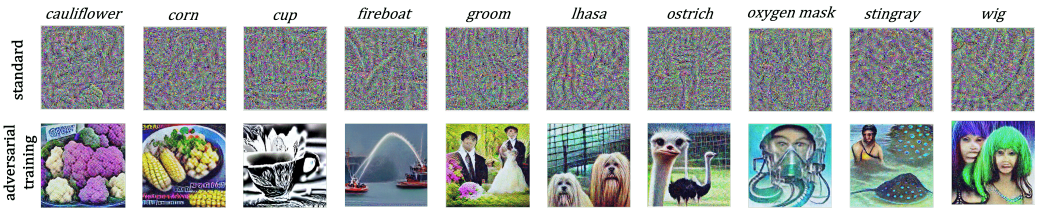


Figure 8.10: Example reconstructions from publicly available pre-trained ResNet50 ImageNet models for both standard and adversarial training. The column labels indicate the ImageNet class that was targeted for reconstruction. **(top)** standard model available from Torchvision.[2] **(bottom)** adversarially trained model available from MadryLab.[3]

This insight draws connections from previous work, which has shown a connection between a model's robustness to adversarial input perturbations and the *interpretability* of its feature representations [37, 60, 65, 148, 154]. Ilyas et al. [65] argue that adversarial examples arise because of "non-robust features" that are reliable predictors (i.e., they are correlated with the class labels), *provided that they remain unperturbed*; i.e., they lose their predictive power under small-norm perturbations. As such, these non-robust features are useful for standard classification, but *not* for robust classification. Additionally, while non-robust features are inherently imperceptible to humans, their "robust feature" counterparts (which *are* useful for robust classification) will tend to be more perceptible, meaning that we might expect robust models to be more likely to learn human-recognizable features.

This reasoning is borne out when we try to probe the behavior of adversarially trained or provably robust models. For example, Tsipras et al. [148] and Etmann et al. [37] find that gradient-based explanations (e.g., [85, 131, 140]) produced on robust models look more intuitive and align more closely with the salient objects in image classification tasks.

In addition to explanations, another technique that is often used to interpret the features learned by a neural network is *feature visualization*, first introduced by Simonyan et al. [131]. Though this has not been made explicit previously in the literature, feature visualizations are essentially an instance of model inversion—prototypical features for a chosen class (or internal neuron) are visualized by finding an input that maximizes the activation corresponding to that class. Interestingly, Tsipras et al. and Helland and VanHoudnos [60] find that feature visualizations on robust models depict recognizable features, unlike those derived on standard-trained models. Figure 8.10 showcases this phenomenon in terms of feature visualizations that we sampled from publicly available pre-trained ImageNet models from Torchvision[2] and MadryLab.[3] [4] While establishing the precise relationship between the visualizations in Figure 8.10 and the training set would require deeper investigation, at a surface level, the reconstructions are suggestive of a number of potential inferences. For example, the text in the visualization of the *corn* class seems to indicate that some training instances of *corn* may have contained text. Artifacts like these immediately raise the question: *Do these types of visualizations leak information about the training data that the model should not have disclosed*?

[2] See `https://pytorch.org/vision`.

[3] See `https://github.com/MadryLab/robustness`.

[4] It should be noted that, although Figure 8.10 shows feature visualizations from a standard model that are clearly not recognizable, it is possible to obtain more recognizable visualizations using more sophisticated techniques—see Figure D.1 in Appendix D.1.

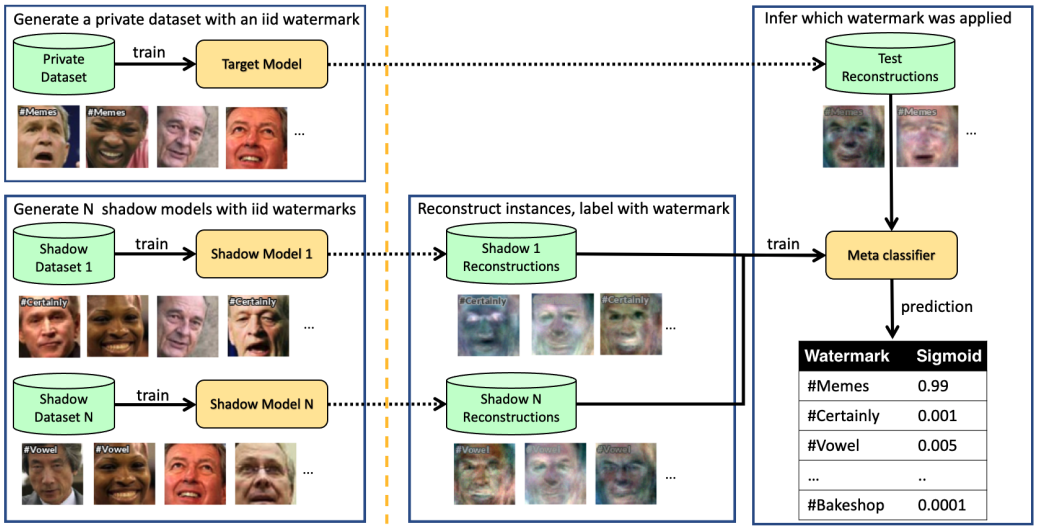


Figure 8.11: Property inference workflow reflecting Algorithm 8.4. The example reconstructions are generated from a dense network trained on LFW-12 with GloRo ($\epsilon = 0.45$) and result in human readable watermarks. Note that watermarks are not always human-readable, but our experiments show that the meta-classifier can give similar results with sufficient training data.

Thus, we argue that increased interpretability has potential privacy implications, when the features learned by the model include protected properties. In such cases, even a naive adversary may come across private information that it had otherwise no reason to even search for. However, to our knowledge, this connection between robustness, interpretability, and disclosure of specific features of a model's training data has not previously been explored.

While there is not necessarily an *a priori* reason that interpretable reconstructions would coincide with training set properties or instances, we posit that if some of the encoded features of the model are overly-specific to protected properties of the training data, private information may be surfaced. Indeed, as we saw in Sections 8.2 and 8.3, deep networks are certainly prone to learning features that compromise privacy. While the features of robust models are more interpretable, and hence less trivially unsound, there is no particular reason to assume they would not be similarly disposed to overfit. In Sections 8.5 and 8.6, we aim to answer the question: *Do these recognizable features found in robust reconstructions reflect specific private properties of the training data?*

8.5 Reconstruction-Based Property Inference Attacks

In this section, we present a novel property inference attack that operates on reconstructions to quantify the extent to which model inversion reveals information about private training properties. Recall from Definition 8.2 that the goal of the property inference adversary is to predict which properties from a known possible set, $\mathcal{P}$, are exhibited by the training set of the adversary's target model. At a high level, we accomplish this by training an *attack model*, g, that takes as input a batch of reconstructions, and outputs a prediction as to which properties would lead to a model producing said reconstructions.

Algorithm 8.4: Property Inference Attack

Inputs: auxiliary information according to Threat Model 8.2: the training procedure $\mathcal{A}$, the possible properties $\mathcal{P}$, and sample access to the distribution $\mathcal{D}$; a number N of shadow models

Output: An attack model, g

def `TrainPiAttack`($\mathcal{A}$, $\mathcal{P}$, $\mathcal{D}$, N):

    // Create N shadow models.
    for $i \in [1, \cdots, N]$ do
        $p^{(i)} \sim$ Uniform($|\mathcal{P}|$)
        $S^{(i)} \sim \mathcal{D}^{\Phi_{p^{(i)}}}$
        $f^{(i)} := \mathcal{A}(S^{(i)})$
    // For each shadow model, produce a
    // batch of reconstructions.
    for $i \in [1, \cdots, N]$ do
        $x'^{(i)} :=$ `GetReconstructions`($f^{(i)}$)
        $z^{(i)} := (x'^{(i)},\ p^{(i)})$
    $T := \left\{ z^{(i)} \text{ for } i \in [1, \cdots, N] \right\}$
    // Train an attack model on the labeled
    // reconstructions.
    $g :=$ `Fit`(T)
    return g

Inputs: the trained attack model g, auxiliary information according to Threat Model 8.2: the model f

Output: A boolean vector of size $|\mathcal{P}|$

def `DoPropertyInference`(g , f):

    $x' :=$ `GetReconstructions`(f)
    return $g(x')$

Our attack is given by Algorithm 8.4 and illustrated with an example workflow in Figure 8.11. Broadly, the attack follows a similar *shadow model* approach first introduced by Shokri et al. [128] in the context of membership inference and later refined for property inference by Ganju et al. [45]. Our attack differs from prior work by using images reconstructed from the model weights (under either a white-box or black-box threat model) instead of the model weights themselves, allowing for significantly fewer shadow models to be used relative to prior work such as [45].

Algorithm 8.4 has two functions: `TrainPiAttack`, which trains the meta classifier g from image reconstructions generated from shadow models, and `DoPropertyInference`, which uses the meta classifier g to predict which properties within the set $\mathcal{P}$ are present in the the target model f.

`TrainPiAttack` takes four inputs: sample access to the data distribution $\mathcal{D}$, the set of possible properties that could be applied to the data $\mathcal{P}$, the training procedure $\mathcal{A}$ used to produce trained models, and N, a hyperparameter specifying the number of shadow models to produce. The first loop of the function generates the N shadow models in turn: for the

i^{th} model, a single property $p^{(i)}$ is uniformly drawn, the property is then applied to the data distribution $\mathcal{D}$ to produce shadow training data $S^{(i)}$ containing that property, and the shadow model $f^{(i)}$ is trained using the training procedure $\mathcal{A}$. The next loop generates the meta classifier's training set T, where the i^{th} element of a T is a collection of reconstructions $x'^{(i)}$ from shadow model $f^{(i)}$ each labeled with the matching $p^{(i)}$ property. Finally, the meta classifier g is trained on T.

`DoPropertyInference` takes two inputs: the meta classifier g and the target model f. Reconstructions are generated from f, passed to the meta-classifier g, and the output is returned.

Figure 8.11 displays an example workflow where $\mathcal{D}$ is LFW-12, $\mathcal{P}$ are the 30 text watermarks from in Figure 8.12, and the training procedure, $\mathcal{A}$, is GloRo (see Chapter 6) with an ℓ_2-adversary whose radius is $\epsilon = 0.45$. Note that the reconstructions in this example produce human-readable watermarks, with "#Memes," "#Certainly," and "#Vowel" being visible in several of the reconstructions.

8.5.1 Private Property Leakage in Reconstructions

We now present experimental evidence to support the following claims: (1) feature reconstruction visualizations derived from deep networks leak specific, non-predictive properties of training data; and (2) while this is true for both non-robust and robust models, it manifests differently in robust models, leaving them more susceptible to a less-powerful attacker.

Experimental Setup

We begin by presenting details on the datasets, watermarks, target models, attack model, and general methodology used for our evaluation.

Datasets. We focus on two datasets in our evaluation: LFW [63] and CIFAR-10 [76]. Both are common, publicly-available benchmark datasets that have been frequently studied in the machine learning privacy literature. We select only images from classes in the full LFW dataset that have at least 50 examples, leaving 12 remaining classes. We refer to this subset of LFW as LFW-12. We use the 3-channel color version of the dataset and crop and scale the images to be 64×64 pixels. The images are then zero-centered and normalized to the unit ball, by dividing by their ℓ_2 norm. We use a single random 80-20 stratified train/test split for all experiments, giving 912 images in the training set and 228 in the test set. For CIFAR-10, we use the standard train and test sets of 50,000 and 10,000 32×32 color images respectively. The images are standardized using Imagenet channel-wise means and variances.

Watermarks. For our watermarking scheme, we randomly generate 30 textual watermarks using Diceware passphrases, from which we—for each experiment—randomly sample 14 (without replacement) to comprise our set of possible properties, $\mathcal{P}$. These watermarks are shown in Figure 8.12. All watermarks use the same white color-scheme with a black border (see Figure 8.2 for an example). We apply watermarks to 80% of the training instances as a pre-processing step; i.e., when training a given model, the same instances will contain watermarks each time they appear in a batch.

#Space #Characteristic #Sentence #Voice #Month #Squiggly #Ongoing #Journey
#Graph #Variety #Poet #Evergreen #Calendar #Brought #Robust #Describe
#Cruncher #Memes #Getting #Certainly #Water #Tray #Bakeshop #Party #Tusk
#Subject #Vowel #Metal #Cloud #Dozen

Figure 8.12: Universe of possible watermarks used in our experiments. We randomly subsample (without replacement) 14 of these watermarks for each distinct experiment.

Target Models. We study three different training methods for the target models in our evaluation: standard training, PGD adversarial training [97], and GloRo training [88] for certified ℓ_p-robustness. For both PGD and GloRo training, we consider ℓ_2 adversaries; on LFW-12 we use a perturbation budget of $\epsilon = 0.45$, and on CIFAR-10 we use $\epsilon = 1.0$ for PGD and $\epsilon = 0.141$ for GloRo. We focus our evaluation on two simple dense architectures respectively containing two and four hidden layers of 1,000 neurons each. For standard and PGD models, use ReLU activations, but for reasons discussed in Section 6.3 in Chapter 6, we use MinMax [4] for GloRo models. All models are trained with a batch size of 64 for 30 epochs using the Adam optimizer [75] with a default learning rate of 0.001.

Property Inference Classifier. The architecture of the attacking meta-classifier takes inspiration from the permutation-invariant approach of Ganju et al. [45]—in our case, we enforce permutation invariance over the set of reconstructions from a given model rather than permutation invariance with respect to said model's neuron parameters. We use a single pre-activation ResNet18 [57] as a feature extractor for each image in the set of reconstructions. We then feed the set of reconstructions into a small set transformer [81] to obtain a single output vector, which we subsequently feed into a linear multi-label classification head to obtain the final prediction. For more details on the set transformer architecture we employ, see Appendix D.2. The ResNet18 feature extractor and set transformer are trained jointly with batches of size 16 (each instance of which is a set of 64 reconstructions) for 100 epochs using Adam with a learning rate of 0.001 and weight decay of 0.01.

Methodology. To implement and evaluate our property inference attack detailed in Algorithm 8.4, we construct multiple different watermarked versions of the same underlying dataset, each of which we train a distinct model over. Over the course of training on any given watermarked dataset, we ensure that a given image retains the same watermark regardless of the batch sampling method—in this sense, our watermarks are static data augmentations.

Trained models are then split into train and test sets for the sake of training and evaluating the attack model. Within the train and test sets, we generate a set of reconstructions for each model, and label the whole set with a multi-label that indicates which watermarks the given model was trained on. For each of our property inference experiments, we fix the number of total watermarks to be $|\mathcal{P}| = 14$.

For every experiment, we perform an 80%-20% train-test split across trained models. From each model, we sample 64 reconstructions where for each reconstruction, we perform 128 iterations of gradient descent on the model inversion objective given by Equation 8.1 with a randomly chosen label $j \sim \text{Categorical}(m)$, where $m = 14$ is the number of classes, each corresponding to a different possible watermark in $\mathcal{P}$. We initialize this optimization process with a blank image, $x_0 = 0$, and ℓ_2-normalize the gradients at each iteration to a step size of 0.001.

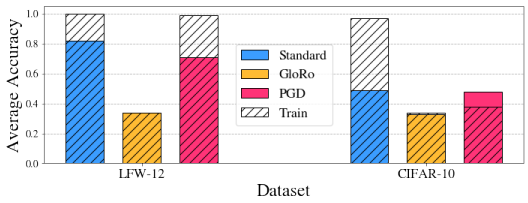
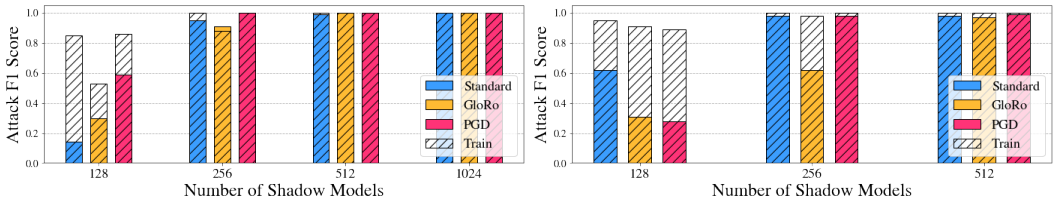


Figure 8.13: The average top-1 accuracy of the shadow models on each dataset. The colored bars indicate test set accuracy, whereas the hatched "Train" bars signify train set accuracy, thereby indicating the generalization gap. We found that the variance in accuracy for each method was negligible.



(a) (b)

Figure 8.14: Results of our property inference attack on the LFW-12 **(a)** and CIFAR-10 **(b)** datasets with varying numbers of shadow models trained. The colors correspond to different shadow model training methods. The solid bars signify F1 score on the test set, while the hatched "Train" bars signify the F1 score on the train set, indicating the generalization gap of the attack.

Hardware. Experiments were performed using an AWS instance with 8 16GB NVIDIA Tesla V100 GPUs and an NVIDIA DGX-2 box with 16 32GB Tesla V100s.

Property Inference Results

Figure 8.13 shows the average performance of the shadow models that we train in terms of top-1 accuracy. We can see that GloRo achieves the worst performance but the smallest generalization gap, which we attribute to the strong regularizing effect that GloRo has on the model's global Lipschitz constant. For PGD, the performance is comparable to standard training, albeit the generalization gap on CIFAR-10 is smaller, which again can be attributed to the regularizing effect of adversarial training.

Figure 8.14a shows the results of our attack in terms of the F1 score against 2-hidden-layer dense models with 1,000 neurons per layer, trained on the LFW-12 dataset as per the previous section. In this experiment, each shadow training set is given one watermark and each image has an 80% chance of receiving this watermark. We can see that increasing the number of shadow models dramatically increases the attack performance, with little improvement past 512 models. With sufficiently many shadow models, all training methods are clearly susceptible to property leakage.

More interestingly, for 128 shadow models, the robust training methods (GloRo and PGD) seem to leak more information relative to standard training. We believe that the large generalization gap is primarily due to the high capacity of our attacking meta-classifier architecture (a stacked pre-activation ResNet18 and set transformer), which can easily overfit.

Figure 8.14b shows the results of our attack against 4-hidden-layer dense models with 1,000 neurons per layer, trained on the CIFAR-10 dataset. Each shadow training set is given one watermark and each image has an 80% chance of receiving this watermark. Due to computational constraints, we did not train an attack using 1024 shadow models. We can see that, similarly to in Figure 8.14a, increasing the number of shadow models yields strong attacking model performance—with a sufficient number of shadow models, the attack was highly successful against all training methods.

Given the strong performance of the property inference attack in a variety of settings, it is clear that reconstructions quantitatively carry information about the distributional properties of the training set, even when those properties do not provide useful information about the model's prediction objective and therefore provide no incentive for memorization.

Visualizing Reconstructions

We now play the role of the weak adversary by inspecting the reconstructions that the property inference attack operates on. While the strong adversary has prior knowledge of the possible watermarks and only needs to distinguish which is which, the weak adversary has no knowledge of the existence of the watermarks unless it is made clearly apparent through the reconstructions.

Following the same methodology as used by the models seen by the property inference adversary, we produced a set of standard, GloRo, and PGD models trained on versions of LFW-12 and CIFAR-10 watermarked with the text, "#privacy." We subsequently produced m reconstructions for each model obtained by performing standard SGD model inversion starting from a gray background. The results for LFW-12 and CIFAR-10 are shown in Figures 8.15 and 8.16, respectively.

For context, we collected the PGD accuracy[5] of each of the models. On LFW, PGD was the most empirically robust with a PGD accuracy (at radius $\epsilon = 0.45$) of 0.61, followed by GloRo at 0.58 and standard at 0.55. On CIFAR-10, GloRo was the most empirically robust with a PGD accuracy (at radius $\epsilon = 1.0$) of 0.33, followed by PGD at 0.21 and standard at 0.20.

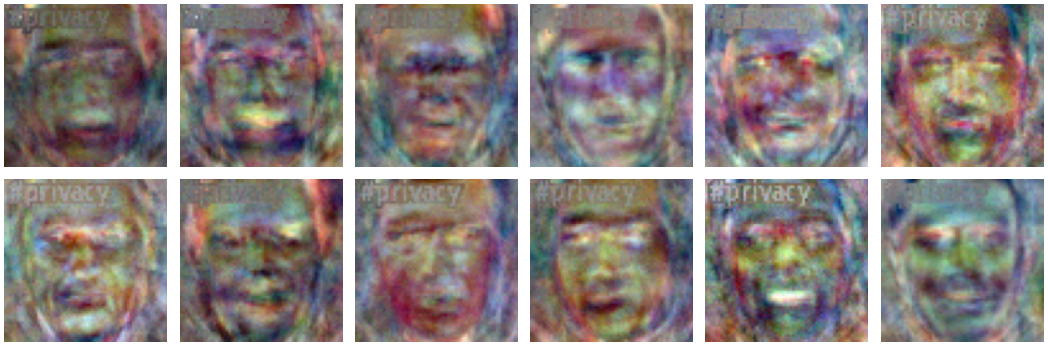
Our primary goal is to look for strong evidence appearing in the reconstructions indicating the presence and nature of the watermark added to the training set. Because the watermarks are fairly small, the resulting figures are best viewed closely.

On LFW the reconstructions obtained on the PGD model appear to show the watermark most clearly; in several of the reconstructions, most notably the one second from the right on the bottom, the text "#privacy" is clearly legible. In most of the other reconstructions the top-left corner of the visualization has a text-like anomaly that at least suggests the presence of *some* watermark. By contrast, the reconstructions on standard models are the least clear. Not only are the faces more vague and indistinct, but with few exceptions the text is difficult to perceive, let alone read. The quality of the reconstructions on the GloRo nets appear to be in-between, with generally clearer text than on the standard model, but perhaps not as clear as on the PGD models. We note that this ordering is consistent with what we would expect given our analysis of previous findings on interpretability; namely the recognizability of the features in the reconstructions—including the private watermark—improves as the models become more robust.

[5]By PGD accuracy, we mean the fraction of test points that are labelled correctly after a perturbation with norm ϵ obtained via PGD is applied. This serves as a measure of the empirical robustness of the model.

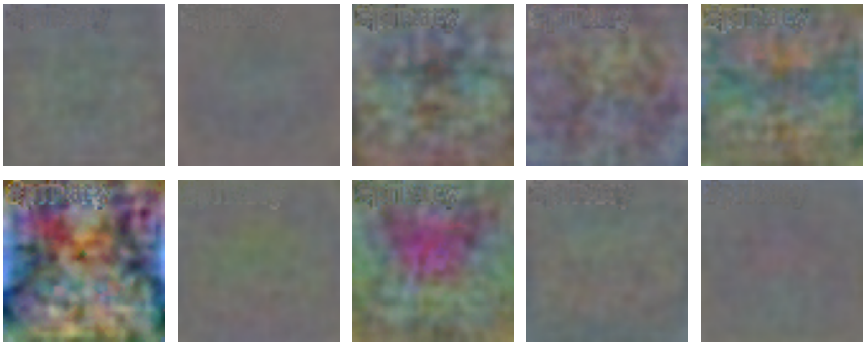
(a) Standard

(b) GloRo



(c) PGD

Figure 8.15: Reconstructions on LFW-12 models obtained using model inversion. Models were trained using **(a)** standard, **(b)** GloRo, and **(c)** PGD adversarial training respectively. Viewed closely, we see that the watermark, "#privacy," can be read in some of the reconstructions, particularly on the robust models.



(a) Standard

(b) GloRo

(c) PGD

Figure 8.16: Reconstructions on CIFAR-10 models obtained using model inversion. Models were trained using **(a)** standard, **(b)** GloRo, and **(c)** PGD adversarial training respectively. Viewed closely, we see that the watermark, "#privacy," can be read in some of the reconstructions, particularly on the robust models.

The results on CIFAR-10 are qualitatively the same. This time, the GloRo model was distinctively the most empirically robust, and as before, we see that this translates to the clearest reconstructions with the most legible watermarks.

Information Organization. Our findings when inspecting reconstructions are in line with recent work that has noticed an increased susceptibility of robust models to privacy attacks. However, despite this, and the fact that the watermarks appear visibly even to a weak adversary, the global success of the strong adversary even in non-robust settings suggests a more nuanced picture. Namely, our results seem to indicate that robust models do not necessarily leak *more* private information than standard models—rather, it is the way in which private information is leaked that differs. The greater success experienced by a relatively weaker adversary on robust models might relate to the overt way in which robust models appear to encode learned information. Meanwhile, a complex ML model, like the meta classifier in our property inference attack, can easily pick up on any sort of signal (when the necessary auxiliary knowledge is available), even if it is virtually imperceptible by some other means.

8.6 Encoding of Private Properties

We can think of an adversary that performs model inversion in Threat Models 8.3 or 8.4 as a *weak* adversary, as compared to a *strong* adversary that performs property inference in Threat Model 8.2. The strong adversary, while perhaps less realistic in practice, serves as a way to quantify the extent to which property information exists in the reconstructions it operates on. On the other hand, the weak adversary informs us what a naive adversary could discover without any special auxiliary knowledge.

As we have seen in Section 8.5, the strong adversary succeeds equally on robust and non-robust models, despite the fact that watermarks are not recovered as plainly on non-robust models as on their robust counterparts. This suggests that the differences between these types of models is perhaps not in the extent to which they leak private information, but rather, the extent to which the information is readily accessible to be exploited *by a weak adversary.*

In fact, we find that in severe cases, the leakage of robust models is so egregious that even entire training instances are encoded directly in hyperplanes determined by the geometry induced by a model's parameters. The remainder of this section will endeavor to provide high-level intuition as to how and why robust models might encode private information in such an overt manner.

8.6.1 Metric Alignment of Robust Features

In their framework for analyzing non-robust features that give rise to adversarial examples, Ilyas et al. [65] consider the learned features of a model to induce a metric on the model's inputs. Adversarial examples arise when this metric is misaligned with the ℓ_p metric used for the adversary's perturbation budget. Conversely, [65, Theorem 2] states that as the adversary's perturbation budget, ϵ, is increased, adversarial training blends the metric induced by the learned features with the adversary's ℓ_p metric. Taken intuitively, this suggests that robust models may tend to learn parameters that roughly compute ℓ_p distances.

Consider how a network might learn to compute the distance from an input, x, to some centroid, c. For the sake of our analysis we will begin by considering a dense network with one hidden layer. The pre-activation output of the first layer is given as $h(x) = xW + b$, where W is an $n \times k$ weight matrix, and b is a bias vector in $\mathbb{R}^k$. Consider the squared ℓ_2 distance from x to c, given by Equation 8.13, which can be rewritten as Equation 8.14.

$$d_{\ell_2}(x, c)^2 = ||x - c||_2^2 \tag{8.13}$$

$$= -2c^T x + ||x||_2^2 + ||c||_2^2 \tag{8.14}$$

According to Equation 8.14, if we set some column vector $w_i \in \mathbb{R}^n$ of W to $-2c$, and we set b_i to $||c||_2^2$, then the i^{th} output of h will be given by $h_i(x) = d_{\ell_2}(x, c)^2 - ||x||_2^2$. If the norm of each data point is roughly the same, i.e., say $||x|| \approx ||x||$ for points in the support of $\mathcal{D}$, then the $||x||_2^2$ term can be incorporated into b_i such that h approximately computes squared ℓ_2 distances to c.

One natural hypothesis then, is that if we train our simple dense network to be robust, some of the features it might encode might correspond to distances to learned centroids, which would be apparent directly from the weights. If the capacity of the network (corresponding to k in our simple case) is large compared to the number of training points, these centroids might correspond to specific training points. In a less over-parameterized setting, the centroids might instead correspond to clusters of points—though this would not preclude them from containing information shared by groups of instances.

Even in networks with more than one hidden layer, we occasionally find that the first-layer weights do indeed contain this sort of information, which can be revealed by simply visualizing the weight columns as an image. However, we do not need to consider only the first layer. To extend our intuition to larger networks we will consider a geometric view of neural networks.

Previous work has observed that ReLU networks divide their input space into a *polyhedral complex*, where the polyhedral regions correspond to *activation patterns* (i.e., which ReLUs are switched on/off) and the boundaries between regions correspond to inputs for which a particular neuron (corresponding to the boundary in question) takes a pre-activation value of zero [44, 69, 119]. For a first-layer neuron, i, the corresponding boundary lies in a single hyperplane with normal vector given by w_i. Thus when we visualize the network's weight columns, we are actually visualizing specific boundaries in the network's polyhedral complex.

We can visualize other boundaries as well. The boundaries of higher-layer neurons do not correspond to a single hyperplane, but they can be visualized with respect to a fixed polyhedral region whose face lies in some n-dimensional hyperplane. In practice, these hyperplanes are easy to compute because they correspond to the input gradients of individual neurons [44].

8.6.2 Uncovering Encodings in Black-box Settings

We return to comment on the black-box threat model (Threat Model 8.4) alluded to in Section 8.1.3. In recent work, Rolnick and Kording [119] introduced an algorithm for faithfully reverse-engineering ReLU networks by analyzing the boundaries of the corresponding polyhedral complex. In brief, the algorithm detects points that intersect with faces in the polyhedral complex via a binary search over a line spanning the input space that detects inflections in the network's output. Hyperplanes corresponding to each face are then approximated by sampling several points from line segments that intersect the hyperplane.

Figure 8.17: Example of explicit instance leakage via the weights of an adversarially-trained dense network. In each pair of images, the weight-based reconstruction is shown on the left, and the most similar training instance (measured by SSIM) is shown on the right for comparison.

In theory, this process can be used to reconstruct a network in its entirety, transforming the black-box setting to a white-box setting. However, as observed previously, in some cases the network may have encoded sensitive information overtly in the boundary hyperplanes, meaning that simply recovering a single offending hyperplane would suffice. This means that the number of queries required for a weak adversary to obtain private information could be substantially less than would be typically required to complete Rolnick and Kording's reverse-engineering procedure. In the following section, we present results obtained using a variant of this approach tailored to our setting.

8.6.3 Property Encoding and Catastrophic Leakage

Let us return to our hypothesis that robust models might be encouraged to overtly encode training set information in their induced geometry, which would become apparent upon inspection of the model's weights. Here, we test this intuition by directly visualizing the first-layer weights in an adversarially-trained dense network.

Our dense architecture contains 1,000 neurons in each hidden layer, meaning that there are 1,000 possible weight columns to visualize in the first layer. To understand which of these contain information traceable to the training set, we find the most similar training instance (measured by SSIM) to each of the weight columns, and sort the resulting pairs of weight columns and training instances by similarity. We then visualize the best matches side-by-side. For this experiment we produced a model with PGD adversarial training on LFW-12 *without watermarks*. The weight-based reconstructions that were most similar to training points are shown with the corresponding training instances in Figure 8.17.

Strikingly, we see a strong resemblance between the reconstructions and entire *specific training instances*, demonstrating a severe privacy violation beyond simply encoding a private watermark. For example, in the first reconstruction on the left, we see a picture of Serena Williams that includes her hand anomalously placed over her mouth.

On the same architecture, we do not see the same phenomenon with CIFAR-10. Presumably this is because on CIFAR-10, which contains 50,000 training instances, our dense architecture is not over-parameterized as it is for LFW which contains just under 1,000 training instances. Thus, according to our intuition from Section 8.6.1, the CIFAR-10 model might instead encode averages of several instances, which would not be meaningful to a human observer in the same way.

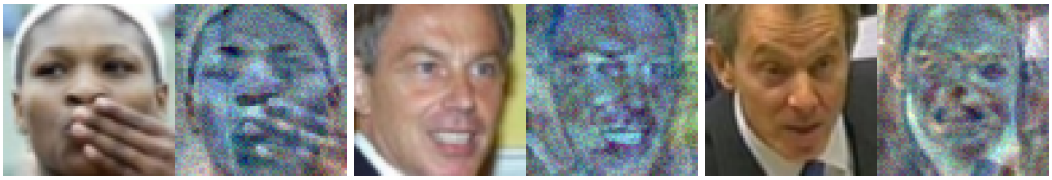


Figure 8.18: Black-box reconstructions derived from weight matrix columns recovered via a black-box model extraction attack derived from Rolnick and Kording [119]. In each pair of images, the reconstruction is shown on the right, and the most similar training instance (measured by SSIM) is shown on the left for comparison.

Black-box Variants

In Section 8.6.2 we also pointed out that reconstructions would be possible in a black-box setting by adapting a network reverse-engineering procedure from prior work [119]. Figure 8.18 shows results obtained in the same way as Figure 8.17, except that the reconstructions are derived from the reverse-engineered weights, rather than the model weights themselves.

We see that the reverse-engineered reconstructions are visually almost indistinguishable from the original reconstructions, leading to many of the same matches. This bears out quantitatively as well; the average cosine similarity between the original weight columns and the reverse-engineered weight columns is 0.98. These results demonstrate that fundamentally, a black-box adversary is no less capable at uncovering overtly-leaked private information. While the black-box adversary requires many queries to the target model this limitation is not as severe as might be expected; though it required roughly 5.5 million queries to reconstruct the entire model, the adversary only needs to come across one offending hyperplane, which can be obtained in roughly 1,600 queries per attempt.

8.7 Related Work

Membership inference

Early membership inference attacks predate the rise of deep learning, and originally targeted statistical summaries. Homer et al. [62] proposed what is considered the first membership inference attack on genomic data in 2008. Since then, a number of studies [52, 123, 130, 152] have looked into membership attacks on statistics commonly published in genome-wide association studies.

More recently, membership inference attacks have been applied to machine learning models. While some of this work has focused on classic machine learning algorithms, such as support vector machines (SVMs) and hidden Markov models (HMMs) [5], as deep learning has become increasingly ubiquitous, membership inference attacks have been particularly directed at deep neural networks. An array of recent efforts [94, 121, 128, 161] have taken varying approaches to membership inference against deep networks in a standard supervised learning setting. Additionally, membership inference has been studied against generative adversarial networks (GANs) [54], and in the context of collaborative, or federated, learning [61, 101].

Black-box attacks We study membership inference as it applies to deep networks in classic supervised learning problems. Most of the prior work in this area [93, 94, 121, 128, 161] has used the *black-box* threat model. Yeom et al. [161] showed that generalization error necessarily leads to membership vulnerability; a natural consequence of this is that a simple "naive" attack (naive), which predicts a point is a member if and only if it was classified correctly, can be found to be quite effective on models that overfit to a large degree. Other approaches have leveraged not only the predictions of the model, but the confidence outputs.

A particularly canonical approach, along these lines, is the attack introduced by Shokri et al. [128] (shadow-bb). In this approach, a the auxiliary data, $\tilde{S}$, is partitioned into two parts, $\tilde{S}_{\text{in}}$ and $\tilde{S}_{\text{out}}$. A *shadow model* is trained on $\tilde{S}_{\text{in}}$, and its outputs on $\tilde{S}_{\text{in}}$ and $\tilde{S}_{\text{out}}$ are used to produce Y_{in} and Y_{out}, respectively. Next, the *attack model* is trained on Y_{in} (labeled 1, indicating Y_{in} represents the shadow model's output on points it saw during training), and Y_{out} (labeled 0). The attack model in the shadow model approach learns to leverage the disparity in prediction confidences on training instances the target model has overfit to, and has been shown to be successful at membership inference on models that have sufficiently high generalization error. A few other membership inference approaches [54, 121] have made use of this same technique. In addition our property inference attack also takes inspiration from this approach.

Despite the fact that shadow model attacks leverage more information than the naive attack, we find in our evaluation (Section 6.4) that often, the shadow model attack fails to outperform the naive attack. One potential reason for this finding is that the learned attack model used by this approach may itself be subject to overfitting. This may be especially true if the attack model picks up on behavior particular to one of the shadow models rather than the true target model. Furthermore, the confidence and entropy of the target model's softmax output is likely to be closely related to whether the target model's prediction was correct or not, meaning that the softmax outputs may not provide substantially different information from that used by naive.

White-box attacks Intuitively, while some information is leaked via the behavior of a model, the details of the structure and the parameters of the model are clear culprits for information leakage. However, few, if any, prior approaches have successfully leveraged this extra information. While Hayes et al. [54] describe a "white-box" attack in their work on membership inference attacks applied to generative adversarial networks (GANs), the attack uses access only the outputs of the discriminator portion of the GAN, rather than the learned weights of either the discriminator or the generator; thus their approach is not white-box in the same sense. Meanwhile, Nasr et al. [107] demonstrated that a simple extension of the black-box shadow model approach to utilize internal activations does not result in higher membership inference accuracies than the original black-box approach. This is perhaps unsurprising, as the internal units of the shadow models are not likely to have any relation to those of the target model (see Section 8.3).

While Nasr et al. proposed an alternative white-box attack that additionally leverages the gradients of the target model's loss function with respect to its weights (which SGD approximately brings to zero on the training points at convergence), in contrast to our work, this attack required a further relaxed threat model, in which the attacker has access to as much as *half of the target model's training data*. The reliance on the target model's training set suggests that Nasr et al.'s attack simply learns differences between the training and auxiliary data, without offering any general insights as to how those differences manifest in the model. Our approach does not require this extra knowledge for the attacker, utilizing

a more restrictive threat model, under which, to our knowledge, no other effective white-box attacks have been proposed.

Robustness, Interpretability, and Privacy

As discussed in Section 8.4, existing literature has drawn the connection between robustness and model interpretability. Much of this work comes from the ML explainability literature, and has focused on demonstrating and justifying the observation that gradient-based explanations on robust models are more intuitive and better align with salient objects in image classifiers [37, 65, 154]. It has also been demonstrated that feature visualizations on robust models are more recognizable than on non-robust models [60, 148]. However, our work is the first to make the connection between these observations and neural network *privacy* explicit. Specifically, we make the observation that feature visualizations are a form of model inversion, and subsequently show that attributes of reconstructions on robust models can be quantitatively and qualitatively linked to private attributes of the model's training set.

Other work by Yeom et al. [162] has separately found that robust networks may be more susceptible to certain kinds of membership inference attacks. These findings complement ours, however, they primarily implicate the fact that the robustness of adversarially-trained models generalizes poorly. That is, adversarially-trained models are more robust on points from their training set than on previously unseen points, while standard-trained models tend to be robust on *neither*. Thus, robust loss serves as a good indicator of membership on adversarially-trained models, but not on their standard-trained counterparts.

On the other hand, our work points to a different source of vulnerability, related to the types of the features learned by robust models, and the way those features are encoded. Although we focus on property rather than membership inference, our findings, in contrast to those of Yeom et al., ultimately indicate that robust models do not necessarily leak *more* private information than standard models—rather, *it is the way in which private information is leaked that differs*. This difference can be reconciled with the results of Yeom et al. with the latter subtle point. Namely, the adversary of Yeom et al. can be considered *weak* in the sense that it relies primarily on only robust loss as an indicator of membership; meanwhile, it is possible that a stronger adversary might be able to use more information to perform equally well on non-robust models for which robust loss is a poor predictor of membership.

Property Privacy

While privacy is often associated with features of specific data points, the privacy literature has also investigated the privacy of properties that apply to groups of training instances or even the training data as a whole. Property privacy is most often studied though property inference attacks like the one presented in Section 8.5, which have been studied in the past [5, 20, 45, 102].

Because it focuses on distributional rather than instance-specific attributes, property inference is typically considered beyond the scope of canonical differential privacy (DP) a rigorous notion of privacy developed by Dwork et al. [34]. Although DP gracefully degrades when applied to groups of correlated records [34, Theorem 2.2], this analysis is typically too stringent to be useful for large sub-populations of an entire dataset.

Without a direct connection to a strong privacy notion such as DP, treating dataset properties as private may come under question. Specifically, prior work on property inference attacks typically lack control with respect to the association between the property of interest and the predictive task itself [5, 20, 45]. It is difficult to claim that a model should *not* reveal

the target property in scenarios with clear association (e.g. gender and income prediction), as there is not a sufficient distinction between these properties and the premises of statistical learning in the first place. After all, we *expect* our model to learn general features that are relevant to its learning objective.

Some previous attempts have been made to address this concern, however. Melis et al. [102] perform property inference with respect to properties that empirically have low correlation with the predictive task (e.g. the presence of glasses in a gender classifier), articulated via the Pearson correlation coefficient. However, it is not clear that small Pearson correlation is sufficient to guarantee a total lack of association and, moreover, is not sufficient to make the assertion that complex models should have no cause to internalize such intuitively irrelevant properties. On the other hand, our study of synthetic watermarks allows us to justify the claim that our target properties should be expected not to be disclosed, and thus to study the extent to which models encode irrelevant features. Specifically, because our experimental setup grants us control over the generative process by which properties occur in the training data, we can be sure that the properties in question are truly independent of the learning task.

Reconstructions

The seminal work on model inversion is by Fredrikson et al. [43]; it showed that simple dense models trained on the AT&T Faces dataset can be inverted to leak personally identifiable reconstructions of training instances. However, Fredrikson et al. focus on reconstructing individuals in the training set, whereas we focus on reconstructing information pertaining to distributional properties of the training data for our reconstruction-based property inference attack. Moreover, not all of our reconstruction attacks are performed by solving a model inversion-esque optimization problem by maximizing class confidence scores—rather, we also perform hyperplane attacks that overtly exploit the geometry induced by ReLU activations and target particular neurons.

Other work [61, 102, 121] has considered GAN-based reconstruction attacks that reveal both distributional properties of and instance level information about the training data. However, these works consider collaborative and online learning scenarios that are not directly comparable to our traditional classification setting. Moreover, these works all require auxiliary generative models to compute reconstructions, whereas we sample reconstructions directly from the classifier itself.

Chapter 9

Feature-Wise Bias Amplification

Bias amplification occurs when the distribution over prediction outputs is skewed in comparison to the prior distribution of the prediction target. For example, in a two-class prediction task where 60% of instances belong to class A and 40% to class B, one would expect a model's overall predictions to match this 60-40 split. However, we would say that the model exhibits bias amplification if its overall predictions tended towards, e.g., 75% class A and 25% class B.

This phenomenon has been observed in several contexts with many of the more recent examples appearing in the algorithmic fairness literature [12, 18, 139, 171], as deep networks that appear to stereotype have been interpreted by some as a possible social issue—in addition to the more clear implications regarding the conceptual soundness of such models. For example, Zhao et al. [171] found that a model tasked with identifying the agent in images depicting various actions was five times more likely to identify the agent as a woman when the corresponding action was cooking, while women were only twice as likely to be the agent in such scenes in the training set. While this now-classic example is perhaps rather innocuous—despite the clear defect in the model's decision process that it illuminates—one could imagine that a model relying heavily on demographic prior information may have more serious repercussions in government applications that determine personalized verdicts, where it is essential that subjects are treated on an *individual* basis.

Intuitively, bias amplification is always undesirable, since it necessarily implies imperfect accuracy. In the strictest sense, however, this is not entirely true: as we will see, bias amplification may be *unavoidable* in contexts where the available features are not sufficiently informative. Indeed, from a Bayesian perspective, the prior is a valid signal to factor into decisions made under uncertainty. When a classifier has access only to a limited set of simple features, it is unclear whether the data can even be reasonably separated. For example, the solvency of a loan applicant may not be ascertainable from his or her income and current liabilities, just as the quality of a job applicant may not be determinable from his or her resume. In such cases, a classifier cannot be perfectly accurate, but may be able to maximize expected rewards by generalizing about, e.g., a job applicant's educational institution.

Nevertheless, in practice, bias amplification has been reported in contexts where the features presumably contain sufficient information to classify accurately, suggesting that it can in fact be a detriment to realizing full utility.

Several factors can cause bias amplification in practice. Classifiers trained to minimize empirical risk may not be sufficiently penalized for ignoring minority classes, and thus bias

amplification is often thought to be result of class imbalance in the training data [17]. There are a myriad of empirical investigations of the effects of class imbalance in machine learning and different ways of mitigating these effects [23, 53, 55, 98, 99, 110, 150]. However, as we show through analysis and experiments, bias amplification can arise in cases where the class prior is not severely skewed, and even when it is unbiased. Thus, techniques for dealing with class imbalance alone cannot explain or address all cases of bias amplification.

Specifically, our work identifies a new source of bias that can be attributed to a suboptimal weighting of particular features in the model. We refer to this strictly detrimental form of bias amplification as *feature-wise bias amplification* (Leino et al. [86]). Feature-wise bias amplification constitutes a violation of conceptual soundness, as it implies that the model does not use features appropriately or justifiably.

9.1 Analyzing Bias Amplification in Binary Classifiers

In this section, we provide a formalization of bias amplification for binary classifiers, and show that in some cases it may be unavoidable. Namely, a Bayes-optimal classifier trained on poorly separated data can end up predicting one label *nearly always*, even if the bias in the prior distribution over labels is minimal. While our analysis makes strong generative assumptions, we show that its results hold qualitatively on real data that resemble these assumptions. We begin by formalizing the setting.

We consider the standard binary classification problem of predicting a label $y \in \{0, 1\}$ given features $x \in \mathbb{R}^n$. We assume that data are generated from some unknown distribution $\mathcal{D}$, and we let $p^* = \Pr[y = 1]$ be the prior probability that the label from a randomly selected point iss 1. Without loss of generality, we will assume that $p^* \geqslant 1/2$; i.e., class 1 is the majority class (if a majority class exists). Definition 9.1 formalizes bias amplification in this setting. Intuitively, bias amplification corresponds to the probability that the model predicts class 1 *in excess of the prior* p^*.

Definition 9.1 (Bias Amplification)**.** *Let $F_S : \mathbb{R}^n \to \{0, 1\}$ be a binary classifier trained on a training set, S, drawn i.i.d. from $\mathcal{D}$. The* bias amplification *of F_S on $\mathcal{D}$, written $B_{\mathcal{D}}(F_S)$, is given by*

$$B_{\mathcal{D}}(F_S) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[F_S(x) - y\big]$$

While bias amplification is given as a property of a particular learned model, we can lift this definition to learning rules to characterize rules that are expected to amplify bias on training sets drawn from $\mathcal{D}$. We say that a learning rule is *bias amplifying* (Definition 9.2) on data distribution $\mathcal{D}$ if it tends to produce models that exhibit bias amplification.

Definition 9.2 (Bias Amplifying Rules)**.** *Let $F_S : \mathbb{R}^n \to \{0, 1\}$ be a binary classifier trained according to learning rule, $\mathcal{A}$, on a training set, S, of N samples drawn i.i.d. from $\mathcal{D}$. We say that $\mathcal{A}$ is bias amplifying if*

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^N} \big[B_{\mathcal{D}}(h_S)\big] \;>\; 0$$

Typically, a classifier's main goal is to produce accurate predictions. A learning rule that always produces a classifier with maximal accuracy—i.e., that minimizes the expected number of errors on labeled points $(x, y) \sim \mathcal{D}$—ought to be considered optimal according to this view. However, what if this optimal learning rule is bias amplifying? While this prospect

seems counter-intuitive, we will demonstrate that it is in fact possible. This suggests that bias amplification is *unavoidable* in some cases, in that it may be necessary to achieve optimal accuracy. In the remainder of this section, we prove this surprising result.

Consider a special case of binary classification in which the data are generated as follows: first, the label $y \in \{0, 1\}$ is drawn according to a Bernoulli distribution with parameter p^*; then features are drawn according to a Gaussian distribution with mean μ_y^*, and covariance matrix Σ^* (Equation 9.1).

$$y \sim \text{Bernoulli}(p^*) \quad x \sim \mathcal{N}(\mu_y^*, \Sigma^*) \tag{9.1}$$

We will further assume that Σ^* is a diagonal matrix, meaning that the features of x are conditionally independent given the class label, and thus the Bayes-optimal predictor for this data can be expressed as the logistic classifier h^* given by Equation 9.2 [106].

$$\begin{aligned} h^*(x) &= \frac{1}{1 + \exp(w^T x + b)} \\ \text{where} \quad w &= \Sigma^{*-1}(\mu_1^* - \mu_0^*), \\ b &= -\frac{1}{2}(\mu_1^* - \mu_0^*)^T w + \log\left(\frac{p^*}{1 - p^*}\right) \end{aligned} \tag{9.2}$$

Suppose that we have access to some omniscient learning rule, $\mathcal{A}^*$, that produces h^* on every training set, S. The question of whether or not $\mathcal{A}^*$ is bias amplifying then simply reduces to asking whether or not $B_{\mathcal{D}}(h^*) > 0$. Theorem 9.1 shows that $B_{\mathcal{D}}(h^*)$ is strictly a function of the class prior p^* and the *Mahalanobis distance*, D, between the class means, μ_1^* and μ_0^*. A particular corollary of Theorem 9.1 is that $\mathcal{A}^*$ is bias amplifying if (and only if) $p^* > 0$; thus, when the prior is unbiased, the optimal (for $\mathcal{D}$) model's predictions remain unbiased (Corollary 9.2).

Theorem 9.1. *Let x be distributed according to Equation 9.1, y be Bernoulli with parameter p^*, D be the Mahalanobis distance between the class means μ_0^*, μ_1^*, and $\beta = -D^{-1}\log(p^*/(1-p^*))$. Then the bias amplification of the Bayes-optimal classifier h^* is:*

$$B_{\mathcal{D}}(h^*) = 1 - p^* - (1 - p^*)\Phi\left(\beta + \tfrac{D}{2}\right) - p^*\Phi\left(\beta - \tfrac{D}{2}\right)$$

Proof. Let X, Y be random variables drawn from $\mathcal{D}$, where $\mathcal{D}$ is given according to Equation 9.1. As noted above, the the Bayes-optimal classifier, h^*, can be expressed as a linear weighted sum in terms of parameters w and b as given by Equation 9.2, such that $\Pr[Y = 1 \mid X = x] = h^*(x)$.

As X has covariance Σ^* and mean μ_y (when $Y = y$), The random variable $w^T X$ is a univariate Gaussian with variance $w^T\Sigma^* w$ and mean $w^T\mu_y^*$ when $Y = y$. Thus the probability that h^* predicts class 1 on X, given $Y = y$ (Equation 9.3) can be expressed as Equation 9.4 where Φ is the CDF of the standard normal distribution.

$$\Pr[h^*(X) = 1 \mid Y = y] = \Pr\left[w^T X > -b \mid Y = y\right] \tag{9.3}$$

$$= 1 - \Phi\left(\frac{-b - w^T\mu_y^*}{\sqrt{w^T\Sigma^* w}}\right) \tag{9.4}$$

Next, we notice that the quantity $w^T(\mu_1^* - \mu_0^*) = (\mu_1^* - \mu_0^*)^T\Sigma^{*-1}(\mu_1^* - \mu_0^*)$ is the square of the Mahalanobis distance between the class means, D^2. Thus we obtain Equations 9.5

and 9.6 expressing the numerator in Equation 9.4 in terms of D for when $Y = 0$ and $Y = 1$, respectively. Similarly, we observe that $\sqrt{w^T \Sigma^* w}$ in the denominator in Equation 9.4 is exactly D.

$$\begin{aligned} -b - w^T \mu_0^* &= \frac{1}{2} w^T (\mu_1^* - \mu_0^*) - \log \frac{p^*}{1-p^*} \\ &= -\log\left(\frac{p^*}{1-p^*}\right) + \frac{D^2}{2} \qquad (9.5) \\ -b - w^T \mu_1^* &= -\log\left(\frac{p^*}{1-p^*}\right) - \frac{1}{2} w^T (\mu_1^* - \mu_0^*) \\ &= -\frac{D^2}{2} - \log \frac{p^*}{1-p^*} \qquad (9.6) \end{aligned}$$

Using the law of total probability we obtain (9.7); substituting in D according to our observations above, we obtain (9.8), and substituting in β yields (9.9).

$$\begin{aligned} \Pr[h^*(X) = 1] &= \Pr[h^*(X) = 1 \mid Y = 0] \Pr[Y = 0] + \Pr[h^*(X) = 1 \mid Y = 1] \Pr[Y = 1] \\ &= (1 - p^*)\left(1 - \Phi\left(\frac{-b - w^T \mu_0^*}{\sqrt{w^T \Sigma^* w}}\right)\right) + p^*\left(1 - \Phi\left(\frac{-b - w^T \mu_1^*}{\sqrt{w^T \Sigma^* w}}\right)\right) \qquad (9.7) \\ &= 1 - (1 - p^*) \cdot \Phi\left(\frac{\log\left(\frac{p^*}{1-p^*}\right)}{D} + \frac{D}{2}\right) - p^* \cdot \Phi\left(\frac{\log\left(\frac{p^*}{1-p^*}\right)}{D} - \frac{D}{2}\right) \qquad (9.8) \\ &= 1 - (1 - p^*) \cdot \Phi\left(\beta + \frac{D}{2}\right) - p^* \cdot \Phi\left(\beta - \frac{D}{2}\right) \qquad (9.9) \end{aligned}$$

Finally, by linearity of expectation, Definition 9.1 yields Equation 9.10.

$$B_{\mathcal{D}}(h^*) = \mathbb{E}[h^*(X) - Y] = \Pr[h^*(X) = 1] - p^* \qquad (9.10)$$

Combining Equation 9.10 with Equation 9.9 gives us our desired result. □

Corollary 9.2. *When x is distributed according to Equation 9.1 and $p^* = 1/2$, $B_{\mathcal{D}}(h^*) = 0$.*

Figure 9.1 shows $B_{\mathcal{D}}(h^*)$ as a function of p^* for several values of D. As the means grow closer together, there is less information available to make reliable predictions, and the label prior is used as the more informative signal. Note that $B_{\mathcal{D}}(h^*)$ is bounded by $1/2$, and the critical point corresponds to bias "saturation" where the model essentially always predicts class 1. From this, it becomes clear that the extent to which over-prediction occurs grows rather quickly when the means are moderately close. For example when $p^* = 3/4$ and the class means are separated by distance $1/2$, the classifier will predict $Y = 1$ with probability close to one.

Table 9.2 shows the effect on real data available on the UCI repository classified using Gaussian Naive Bayes (GNB) classification. These datasets were chosen because their distributions roughly correspond to the naive Bayes assumption of conditional feature independence. In each case, bias amplification occurs in approximate correspondence with Theorem 9.1, tracking the empirical class prior and class distance to Figure 9.1.

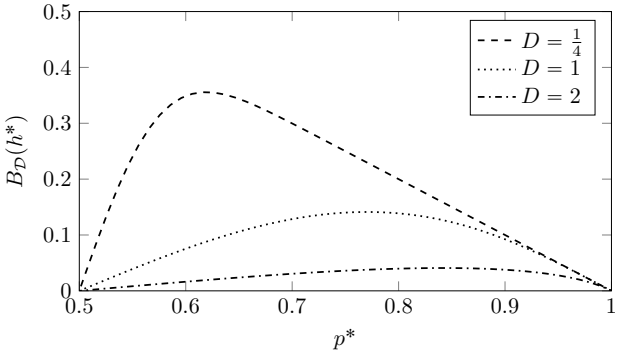


Figure 9.1: Bias amplification of Bayes-optimal classifier as given by Theorem 9.1 in terms of the Mahalanobis distance D between class means and prior probability p^* of observing the label "1."

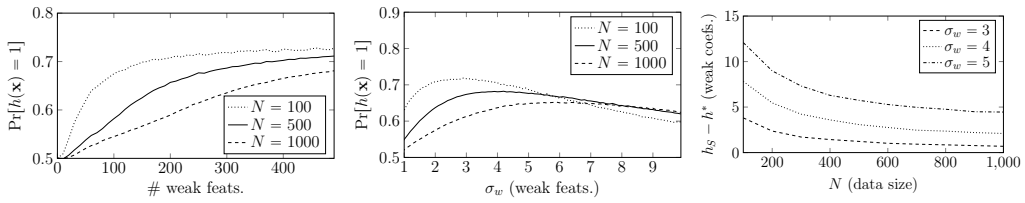
dataset	D	p^*	$B_{\mathcal{D}}(F_S)$	*% acc*
banknote	1.87	0.56	0.04	84.1
breast cancer wisc	1.81	0.63	0.02	94.2
drug consumption	0.86	0.78	0.12	75.6
pima diabetes	1.15	0.66	0.08	79.9

Table 9.2: Bias amplification on real datasets classified using a GNB model.

The results stated by Theorem 9.1 and Corollary 9.2, while perhaps surprising, can be more intuitively understood when taken to their logical ends. Suppose that none of the features carry any information about the label whatsoever—i.e., $\mu_1^* = \mu_0^*$. In such a case, the best we can hope to do for classifying the data accurately is to resort to predicting the majority class on all points. From this reasoning, it is perhaps clear that the cases where bias amplification is unavoidable correspond to those where the features are not sufficiently informative to separate the data by class, leading to uncertainty in our predictions. The less certainty we have, the more valuable the prior is for providing the best estimate of the label.

9.2 Feature Asymmetry and Gradient Descent

When the learning rule does not produce a Bayes-optimal predictor, it may be the case that excess bias can safely be removed without harming accuracy. To support this claim, we begin by turning our attention to logistic regression classifiers trained using stochastic gradient descent (SGD). Logistic regression predictors for data generated according to Equation 9.1 converge in the limit to the same Bayes-optimal predictors studied in Proposition 9.1 and Corollary 9.2 [106]. Meanwhile, such models make fewer assumptions about the data and are therefore more widely-applicable; but as we demonstrate in this section, this flexibility comes at the expense of an inductive bias that can lead to systematic bias in the learned predictors. To show this, we continue under our assumption that x and y are generated according to Equation 9.1, and consider the case where $p^* = 1/2$. In this case, according to Corollary 9.2, any bias amplification that emerges during must come from differences between the trained classifier h_S and the Bayes-optimal h^*.



(a) (b) (c)

Figure 9.3: (a), **(b)**: Expected bias as a function of **(a)** number of weak features and **(b)** variance of the weak features, shown for models trained on $N = 100, 500, 1000$ instances. σ_w in **(a)** is fixed at 10, and in **(b)** the number of features is fixed at 256. **(c)**: Extent of overestimation of weak-feature coefficients in logistic classifiers trained with stochastic gradient descent, in terms of the amount of training data. The vertical axis is the difference in magnitude between the trained coefficient (h_S) and that of the Bayes-optimal predictor (h^*). In **(a)-(c)**, data is generated according to Equation 9.11 with $\sigma_s = 1$, and results are averaged over 100 training runs.

Feature Asymmetry

We will demonstrate empirically that SGD on logistic regressors is a bias-amplifying learning rule when the data exhibit certain forms of feature asymmetries. To define what is meant by "feature asymmetry" in our context, consider the orientation of each feature x_j as given by the sign of $\mu_{1j} - \mu_{0j}$. The sign of each coefficient in h^* will correspond to its feature orientation, so we can think of each feature as being "towards" either class 0 or class 1. Likewise, we can view the full set of features as being *asymmetric towards* y when there are more features oriented towards y than towards $1 - y$.

In addition to the orientation of a feature, we find the relative strength of the feature plays an important role in the observed bias amplification. Informally, we consider a feature to be "weak" if its corresponding influence in the Bayes-optimal predictor is small, and "strong" if its corresponding influence is large. That is, weak features are relatively less important in determining the class label than strong features. In reality, there is no particular numerical cutoff to classify a feature as weak or strong, as indeed there is no strict dichotomy between weak and strong features. Nonetheless, we will use the terms "weak" and "strong" features for illustrative purposes, and our experiments in this section will impose a clear dichotomy between the two.

Our experiments demonstrate that on synthetic data generated in accordance with the naive Bayes assumption (Equation 9.1), logistic regressors learned via SGD exhibit consistent bias towards the class with more weak features oriented in its direction, beyond what is characterized in Theorem 9.1. Specifically, we consistently observe bias amplification even when $p^* = 1/2$. Our analysis from Section 9.1 indicates that in such settings, the optimal predictor exhibits no bias amplification; thus, the bias amplification we observe here cannot be justified on the grounds of improving model utility in the face of uncertainty.

Figure 9.3 explores this phenomenon through synthetic Gaussian data exhibiting the sort of feature asymmetry described above, in which the strongly-predictive features have low variance $\sigma_s = 1$, and the weakly-predictive features have relatively higher variance $\sigma_w > 1$—note that in the Gaussian naive Bayes setting, for a fixed separation between class means, the optimal weight (equal to its influence) of a feature is inversely proportional to its variance. Specifically, the data used here follows Equation 9.1 with the parameters shown in Equation 9.11. Essentially, we maintain an even prior over classes 0 and 1, and provide

one strong feature oriented towards each class, with the remaining features weakly oriented towards class 1. The mean-separation between classes is fixed at 1.0 for each feature,

$$p^* = 1/2 \qquad \Sigma^* = \begin{bmatrix} \sigma_s & & & & \\ & \sigma_s & & & \\ & & \sigma_w & & \\ & & & \ddots & \\ & & & & \sigma_w \end{bmatrix} \qquad (9.11)$$
$$\mu_0^* = (0, 1, 0, \ldots, 0)$$
$$\mu_1^* = (1, 0, 1, \ldots, 1)$$

Figure 9.3a demonstrates that as the disparity in weak features towards class 1 increases, so does the expected bias towards that class. As noted, this bias cannot be explained by Theorem 9.1, because this data is distributed with $p^* = 1/2$. Rather, it is clear that the effect diminishes as the training size increases leading h_S to converge towards h^*.

Figure 9.3b demonstrates that for a fixed disparity in weak features, the features must be sufficiently weak in order to cause bias amplification. This suggests that a feature imbalance alone is not sufficient for causing bias amplification in the learning rule. Moreover, the weak features, rather than the strong features, are primarily responsible for the bias, demonstrated by the fact that the bias amplification decreases σ_w approaches $\sigma_s = 1$. As the training size increases, the amount of variance required to cause bias increases. However, when the weak features have sufficiently high variance, their impact on the bias and accuracy of the model eventually tends to diminish, presumably because the model will ultimately decrease the influence of features that become indistinguishable from noise.

With a finite training sample, the weights learned by any learning rule that converges to the Bayes-optimal predictor will inevitably contain some error. Figure 9.3c plots the average (signed) error between the weak feature weights of h_S and of h^* as the training set size increases. That is, if there are n weak features and the weights of h_S and h^* are $\hat{w}$ and w, respectively, the y-axis of Figure 9.3c plots $\frac{1}{n}\sum_{i=1}^{n} \hat{w}_{i+2} - w_{i+2}$ (note that the indexing of $\hat{w}$ and w reflects the fact that there are two strong features). We see from Figure 9.3c that the weights of the weak features are consistently *overestimated* by SGD prior to convergence, with this effect being more pronounced the weaker the features are and the further we are from convergence. This suggests that the vulnerability of SGD to this phenomenon may be because it is inductively biased towards overestimating the importance of the relatively weaker features. In effect, this means that prior to convergence, the errors in the weights steering the model's predictions will accumulate in favor of the class with more weak features, all else being equal. Thus, because this mode of bias amplification can be traced to the model's incorrect weighting of features, we refer to the phenomenon as "feature-wise" bias amplification.

In addition to the insights provided through our experiments thus far, we further hypothesize that (1) error in the weights of relatively weaker features leads to disproportionately more errors than error in the weights of relatively stronger features, and (2) overestimation of weak features negatively impacts the model more than underestimation (provided the sign of the influence is not flipped).

Feature-wise Bias Amplification in Other Settings

While feature-wise bias amplification may arise when methods other than SGD-based logistic regression are used to produce the model, Figure 9.4 shows that it occurs consistently in models trained using SGD. In particular, we discovered the same bias amplification effect

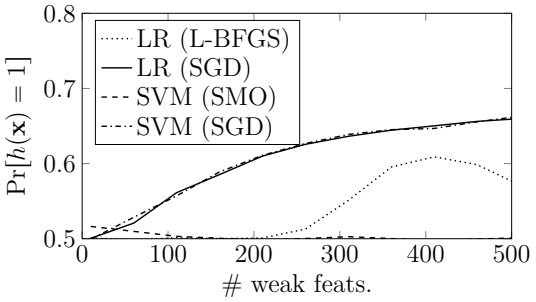


Figure 9.4: Bias from linear classifiers on data generated according to Equation 9.11 with $\sigma_s = 1$ (i.e., generated in the same manner as the experiments in Figure 9.3), averaged over 100 training runs. The SVM trained using SMO used penalty $C = 1.0$ and the linear kernel. Not shown are results for classifiers trained with SGD using modified Huber, squared hinge, and perceptron losses, all of which closely match the two curves shown here for SGD classifiers.

in the asymmetric feature setting when using SGD on multiple different loss functions; this suggests that it is SGD that is bias amplifying, and not logistic loss in particular.

On the other hand, comparable classifiers trained using other learning rules may not necessarily exhibit the same phenomenon. For example, Figure 9.4 shows that logistic regression trained with L-BFGS [92] exhibits less bias amplification than when trained with SGD, and remains unbiased for much larger numbers of weak features. Similarly, support vector machines trained using SMO [115] exhibited essentially no bias amplification in the asymmetric feature regime considered in our analysis. By comparison, support vector machines trained with SGD result in the same bias amplification as logistic regressors.

While we cannot perform the same type of analysis on deep networks (as results regarding the Bayes-optimal predictor for such models are lacking), ...view evidence... that some form of feature-wise bias amplification occurs in deep networks; i.e., the phenomenon is not limited to linear models.

9.3 Mitigating Feature-Wise Bias Amplification

In order to further test the hypothesis that the presence of weak features may lead to bias amplification in models trained with SGD, we consider the implications our analysis from Section 9.2 raises for the mitigation of feature-wise bias amplification. Specifically, we propose that a tailored feature selection procedure that removes weak features will reduce the degree of bias amplification observed. Since our analysis shows that feature-wise bias amplification is fundamentally *avoidable*—in that it is not necessary for maximizing the utility of the model—we should expect that particularly effective mitigation measures should not harm accuracy. If our conjecture from Section 9.2 is indeed correct that underestimating the importance of weak features is less harm than overestimating their performance, removing the weak features—equivalent to underestimating their importance as zero—will have a net positive impact on the model's performance. In the remainder of this section we present experiments that evaluate the extent to which these insights can be validated through their translation to effective mitigation techniques.

dataset	*features*	*instances*
CIFAR-10-binary	2,352	12,000
CelebA-attractiveness	116,412	202,599
Arcene	10,000	200
Colon	2,000	62
Glioma	22,283	85
Micromass	1,300	571
PC/Mac	1,943	3,289
Prostate	5,966	102
Smokers	19,993	187
Synthetic	1,002	100

Table 9.5: Number of features and instances for each dataset used in our evaluation.

Experimental Setup

We begin by describing the datasets, models and mitigation approaches used in our evaluation.

Datasets. We performed experiments over eight tabular binary classification datasets from various domains (rows 3-11 in Table 9.6) and two image classification datasets (CIFAR-10, CelebA). Table 9.5 details the number of features and instances for each dataset. Among the tabular datasets, all were obtained from the scikit-feature repository,[1] except for *micro-mass*, which was obtained from the UCI repository.[2] The synthetic dataset was generated in the manner described by Equation 9.11 in Section 9.2, containing one strongly-predictive feature ($\sigma_s^2 = 1$) for each class, 1,000 weak features ($\sigma_w^2 = 3$), and $p^* = 1/2$. To adhere to the binary setting for which bias amplification was defined, we created a binary classification problem from CIFAR-10 from the "bird" and "frog" classes. We selected these classes as they showed the greatest posterior disparity on a network we trained on the original dataset. For CelebA, we used the *attractiveness* annotation (provided with the dataset) as a binary learning target.

Models. On the tabular datasets, we used simple logistic regression models trained to convergence using SGD. Each experiment was repeated over 20 training runs on a randomly selected stratified train/test split, and the results were averaged. For the image datasets, we used a deep convolutional architecture, VGG-16, to test the extent to which feature-wise bias amplification arises in non-linear networks. We slice deep networks at their penultimate layer, which we interpret as a linear classifier, g on top of a deep feature extractor, h. Thus, when referencing the "features" of a deep model, we refer to the latent representation produced by h.

Mitigation Approaches. We include three mitigation approaches in our evaluation, two of which can be considered post hoc feature selection, and one which is a training regularizer that discourages the learning of weak features. The first feature selection method, which we call the "feature parity" approach, is motivated by the fact that bias amplification may be caused by a disparity in the number of features oriented towards each class. We can attempt to remedy this disparity by enforcing parity in the number of features for each class. To

[1] See `http://featureselection.asu.edu`.
[2] See `http://archive.ics.uci.edu/ml`.

dataset	p^* (%)	*asymm. (%)*	$B_{\mathcal{D}}(h_S)$ (%)	$B_{\mathcal{D}}(h_S)$ (%) *(post-fix)*			*acc. (%)*	*acc. (%) (post-fix)*		
				par	*exp*	ℓ_1		*par*	*exp*	ℓ_1
CIFAR-10-binary	50.0	52.0	1.8	1.7	**0.4**	-	93.0	93.1	**94.0**	-
CelebA-attractiveness	50.4	50.2	7.7	7.7	**0.2**	-	79.6	79.6	**79.9**	-
arcene	56.0	57.7	2.7	**0.6**	1.2	1.7	68.9	69.0	**74.2**	69.4
colon	64.5	51.0	23.1	22.9	**22.6**	35.5	58.5	58.7	58.7	**64.5**
glioma	69.4	54.8	17.4	17.4	**12.2**	17.0	76.3	76.3	**76.7**	75.44
micromass	69.0	54.1	0.68	**0.66**	0.69	0.68	**98.4**	**98.4**	**98.4**	**98.4**
pc/mac	50.5	60.6	1.6	1.6	**1.4**	1.6	**89.0**	**89.0**	88.0	**89.0**
prostate	51.0	44.4	47.3	47.2	**9.8**	28.1	52.7	52.8	**90.2**	71.3
smokers	51.9	50.4	47.4	45.4	**8.0**	33.0	50.0	50.7	**59.0**	51.2
synthetic	50.0	99.9	24.1	17.2	23.6	**5.7**	74.9	**77.9**	74.8	71.4

Table 9.6: Bias measured on real datasets, and results of applying one of three mitigation strategies: feature parity (*par*), experts (*exp*), and ℓ_1 regularization. The columns give: p^*, the class prior for the majority class ($y = 1$); the feature asymmetry (*asymm.*) given as the fraction of features oriented towards $y = 1$; $B_{\mathcal{D}}(h_S)$, the bias amplification of the learned model on test data, which we measure before and after each fix; and the test accuracy, (*acc.*), the test accuracy before and after each fix. The first two rows contain the experiments on deep networks, and the remainder are averaged from 20 training runs of logistic regression optimized via SGD. We did not apply ℓ_1 regularization to the experiments on deep networks.

avoid removing features that are useful for correct predictions, we order the features by their influence on the model's output, removing the least influential features from the class receiving the majority of the features until parity is reached. If the model has a bias term, we adjust it by subtracting the product of each removed coefficient and the mean of its corresponding feature in order to preserve the calibration of the model. We simulate the removal of a feature by simply setting each of its outgoing weights to zero.

The second feature selection approach, which we term the "expert" approach (in reference to an experiment in a similar spirit by Leino et al. [85]), takes the more nuanced view gleaned from Figure 9.3b, which suggests that it is not only the *number* of weak features that matters, but their relative strength as well. Thus, rather than considering the number of features for each class, we can instead remove all features whose strength is below some threshold. We parameterize this approach in terms of two parameters, α and β, where we preserve the α features with the most positive influence and β features with the most negative influence, removing the remaining features which are relatively weaker in their respective directions. In our experiments, we determine α and β by finding the values that minimize the bias amplification on the training data subject to not harming accuracy. We note that this is always possible by selecting all the features, but we did not typically observe this procedure resulting in a degenerate solution.

Finally, we consider ℓ_1 regularization as a possible approach, as it encourages sparsity in the learned weights, and discourages assigning nonzero small-valued weights. However, the intuition of this approach is obfuscated by the fact that it modifies the training procedure itself.

Evaluation Results

Our results are shown in Table 9.6. To summarize, on every dataset we consider, at least one of the proposed mitigation methods proves effective at reducing the classifier's bias amplification. In all but two cases, the expert method showed the best performance in

terms of bias amplification removal, and was able to reduce bias amplification in all but one case. In terms of accuracy, our feature selection approaches nearly always improve classifier performance—in some cases significantly.

Notably, in each case where feature parity removed bias amplification, the accuracy was likewise improved, supporting our claim that bias resulting from asymmetric feature regimes is avoidable. However, in most cases, the benefit from applying feature parity is rather small. The *arcene* dataset is the exception, which is likely due to the fact that it has large feature asymmetry in the original model, leaving ample opportunity for improvement through the feature parity approach.

On the other hand, the results in Table 9.6 suggest that the expert approach is the most effective mitigation technique, both in terms of bias removal and accuracy improvement. In most datasets, this approach reduced bias while improving accuracy, often substantially. This effect is best exemplified on the *prostate* dataset, where the experts approach removed 80% of the bias amplification while improving accuracy from 57.7% to 90.2%. Similarly, for *arcene* and *smokers*, experts removed over 50% of the bias amplification while improving accuracy by 5 and 9 percentage points respectively.

Meanwhile, ℓ_1 regularization proved least reliable at removing bias amplification subject to not harming accuracy. In many cases, it was unable to remove much bias amplification (*glioma, micromass, PC/Mac*). Though it performed better on several real datasets (*arcene, prostate, smokers*), even removing up to 40% of the bias amplification on the *prostate* dataset, it was consistently outperformed by either the parity or expert method. Additionally, on the *colon* dataset, it made bias amplification significantly worse.

The feature orientations observed on CIFAR-10-binary and CelebA-attractiveness reveal that deep networks tend to compute features with less asymmetry than exhibited by the tabular data in our evaluation; we expect this to render the feature parity approach less effective on deep networks. Our mitigation results confirm this, although on CIFAR-10-binary, enforcing feature parity had some effect on bias amplification and a proportional positive effect on accuracy. The experts approach, on the other hand, continued to perform well for the deep models. While this approach generally had a greater effect on accuracy than bias for the linear models, this trend reversed for deep networks, where the decrease in bias amplification was consistently greater than the increase in accuracy. For example, the 7.7% of excessive bias in the original CelebA-attractiveness model was reduced by approximately 98% to 0.2%, effectively eliminating it from the model's predictions. The overall effect on accuracy remained modest (0.3% improvement).

These results on deep networks may be somewhat surprising, considering that our analysis is driven by observations derived on simple linear classifiers. While the improvements in accuracy are not as significant as those seen on linear classifiers, they align with our expectations regarding bias reduction in linear models. This suggests that future work might improve on these results by adapting the analysis and approaches described in this chapter to better suit deep networks.

Chapter 10

Conclusion

With the rapidly growing success of deep networks at tasks accessible only to humans not long ago, it is sometimes easy to forget that "artificial intelligence" is far from solved. As we have explored in this thesis, despite their remarkable progress over the years, neural networks have several peculiar shortcomings and vulnerabilities. For example, we have seen that deep networks are easily fooled by small, meaningless perturbations (Part II), and may leak sensitive information about the data they are trained on (Part III). These weaknesses serve as a reminder that the success of deep networks remains shallow compared to the lofty goal of truly "intelligent" machines.

When neural networks appear to match human performance in some problem setting, it is natural to ask how their success comes about. Does the success of neural networks at human tasks imply that both utilize similar processes? In many tasks, this would be the ideal; some indication of convergence between human and machine logic would give us confidence in a network's ability to handle inputs it was not explicitly tested on. *Conceptual soundness* (Chapter 2) captures this sentiment—when the model encodes and uses features that agree with human intuition, it is easier to believe its good behavior will generalize.

By contrast, a model may learn completely different ways of accomplishing its objectives—i.e., ways that are not conceptually sound. It is natural to ask though, *is this a problem*? Human and machine reasoning may simply be different by nature—is *our* way necessarily better? In response to such lines of inquiry, the work in this thesis ultimately suggests that conceptually unsound behavior lies at the heart of many of the weaknesses that cause modern machine learning to fall short of "human-level" performance (despite occasional claims to the contrary). These findings give rightful pause to any inclination towards accepting opaque machinery that cannot be corroborated by human intuition.

Well designed *explanations* (Chapter 3) allow us to evaluate the degree to which learned models are conceptually sound. Crucially, explanations should allow us to distinguish between models that are complex, yet conceptually sound, and those that are unintelligible and brittle (see Chapter 4 for further discussion). To this end, an objective look at the logic and encodings employed by typical deep networks reveals the truth—that modern deep networks are seldom conceptually sound. To remedy this, we must turn to improving the quality of the underlying models.

One major barrier to obtaining conceptual soundness is contemporary learning algorithms' proclivity to encode *non-robust features* (see Chapter 5) as representational "shortcuts" for optimizing their learning objectives. In addition to the complications this causes

for conceptual soundness, such features compromise the security of the resulting network, as they lead to *adversarial examples* (see Chapter 5). Both consequences can be mitigated through a *robust* learning objective, which precludes small-norm adversarial examples and the non-robust features that give rise to them. With aid from our contributions, robust training is becoming more effective and scalable, and we anticipate continued effort in the directions laid out in our work from Chapters 6 and 7 will bring continued progress to larger and more challenging domains in the foreseeable future. Still, far from being solved, extending robust learning to "semantic" attacks is an outstanding challenge with less tangible solutions (see Chapter 7 for further discussion).

Finally, the work in the third part of this thesis reveals that robustness to malicious input perturbations is only the first step in achieving trustworthy "human-like" intelligence. Specifically, we uncover several orthogonal weaknesses and vulnerabilities relating to the conceptual soundness of deep networks, affecting both model privacy (Chapter 8) and calibration (Chapter 9). The insights in our work illuminate key mechanisms behind unsound network behavior; at the same time, we expect they will serve as important guidance for improving conceptual soundness in the neural networks of the future.

Appendix

Appendix A

Further Details on GloRo Nets

A.1 Tighter Bounds for Theorem 6.1

Note that in the formulation of GloRo Nets given in Section 6.1, we assume that the predicted class, j, will decrease by the maximum amount within the ϵ-ball, while all other classes increase by their respective maximum amounts. This is a conservative assumption that guarantees local robustness; however, in practice, we can dispose of this assumption by instead calculating the Lipschitz constant of the margin by which the logit of the predicted class surpasses the other logits, $f_j - f_i$.

The *margin Lipschitz constant* of f, defined for a pair of classes, $i \neq j$, is given by Definition A.1.

Definition A.1. *Margin Lipschitz Constant For network, $f : \mathbb{R}^n \to \mathbb{R}^m$, and classes $i \neq j \in [m]$, K^*_{ij} is an upper bound on the margin Lipschitz constant of f if $\forall x_1, x_2$,*

$$\frac{|f_j(x_1) - f_i(x_1) - (f_j(x_2) - f_i(x_2))|}{||x_1 - x_2||} \leqslant K^*_{ij}$$

We now define a variant of GloRo Nets as follows: For input, x, let $j = F(x)$, i.e., j is the label assigned by the underlying model to be instrumented. Define $\bar{f}^\epsilon_i(x) ::= f_i(x)$, and $\bar{f}^\epsilon_\perp(x) ::= \max_{i \neq j}\{f_i(x) + \epsilon K^*_{ij}\}$.

Theorem A.1. *Under this variant, if $\bar{F}^\epsilon(x) \neq \perp$, then $\bar{F}^\epsilon(x) = F(x)$ and F is ϵ-locally-robust at x.*

Proof. The proof is similar to the proof of Theorem 6.1 (Section 6.1). Let $j = F(x)$. As before, when $\bar{F}^\epsilon(x) \neq \perp$, we see that $\bar{F}^\epsilon(x) = j = F(x)$.

Now assume x' satisfies $||x - x'|| \leqslant \epsilon$. Let K^*_{ij} be an upper bound on the margin Lipschitz constant. Then, $\forall i$

$$|f_j(x) - f_i(x) - (f_j(x') - f_i(x'))| \leqslant K^*_{ij}\epsilon \tag{A.1}$$

We proceed to show that for any such x', $F(x')$ is also j. In other words, $\forall i \neq j$, $f_i(x') < f_j(x')$. By applying (A.1), we obtain (A.2). Next, (A.3) follows from the fact that $\bar{f}^\epsilon_\perp(x) = \max_{i \neq j}\left\{f_i(x) + K^*_{ij}\epsilon\right\}$. We then obtain (A.4) from the fact that $f_j(x) > \bar{f}^\epsilon_\perp(x)$, as

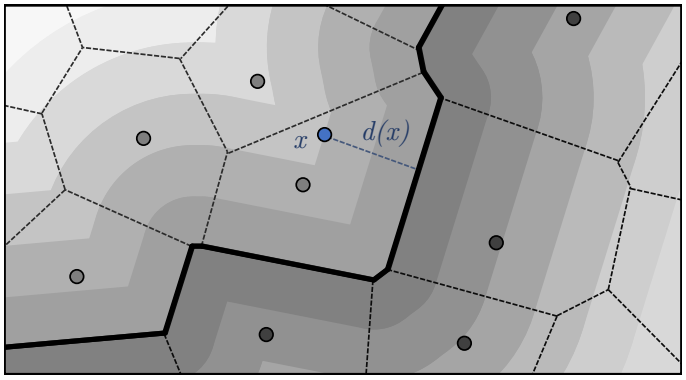


Figure A.1: Illustration of a function, g, constructed to satisfy Theorem 6.3. The points in S are shown in light and dark gray, with different shades indicating different labels. The Voronoi tessellation is outlined in black, and the faces belonging to the decision boundary are highlighted in bold. The level curves of g are shown in various shades of gray and correspond to points, x, at some fixed distance, $d(x)$, from the decision boundary.

$\bar{F}^{\epsilon}(x) = j \neq \perp$.

$$\begin{aligned}
& \cancel{f_i(x)} + f_j(x) - \cancel{f_i(x)} - f_j(x') + f_i(x') \\
\leqslant\ & f_i(x) + |f_j(x) - f_i(x) - (f_j(x') - f_i(x'))| \\
\leqslant\ & f_i(x) + K^*_{ij}\epsilon && \text{(A.2)} \\
\leqslant\ & \bar{f}^{\epsilon}_{\perp}(x) && \text{(A.3)} \\
<\ & f_j(x) && \text{(A.4)}
\end{aligned}$$

Rearranging terms, we obtain that $f_i(x') < f_j(x')$. Thus, $F(x') = j$; this means that F is locally robust at x. □

A.2 Proof of Theorem 6.3

Theorem. *Let f be a binary classifier that predicts $1 \iff f(x) > 0$. Let $K_L(x, \epsilon)$ be the local Lipschitz constant of f at point x with radius ϵ.*

Suppose that for some finite set of points, S, $\forall x \in S$, $|f(x)| > \epsilon K_L(x, \epsilon)$, i.e., all points in S can be verified via the local Lipschitz constant.

Then there exists a classifier, g, with global Lipschitz constant K_G, such that $\forall x \in S$, (1) g makes the same predictions as f on S, and (2) $|g(x)| > \epsilon K_G$, i.e., all points in S can be verified via the global Lipschitz constant.

Proof. Let T be the Voronoi tessellation generated by the points in S. Each Voronoi cell, $C_j \in T$, corresponds to the set of points that are closer to $p_j \in S$ than to any other point in S; and the face, $F_{ij} \in T$, which separates cells C_i and C_j, corresponds to the set of points that are equidistant from p_i and p_j.

Let $B = \{F_{ij} : \text{sign}(f(p_i)) \neq \text{sign}(f(p_j))\}$, i.e., the set of faces in the Voronoi tessellation that separate points that are classified differently by f (note that B corresponds to the boundary of the 1-nearest-neighbor classifier for the points in S).

Consider a point, x. Let $p_x \in S$ be the closest point in S to x, i.e., the point corresponding to the Voronoi cell containing x. Let $d(x) = ||\text{proj}(x \to B) - x||$; that is, $d(x)$ is the minimum distance from x to any point in any of the faces in B. Then define

$$g(x) = \text{sign}\big(f(p_x)\big)\frac{d(x)}{\epsilon}$$

First, observe that $g(x) > 0 \iff f(x) > 0$ follows from the fact that $d(x)$ and ϵ are non-negative, thus the sign of $g(x)$ is derived from the sign of $f(x)$.

Next, we show that the global Lipschitz constant of g, K_G, is at most $1/\epsilon$, that is, $\forall x_1, x_2$,

$$\frac{|g(x_1) - g(x_2)|}{||x_1 - x_2||} \leqslant \frac{1}{\epsilon}$$

Consider two points, x_1 and x_2, and let p_1 and p_2 be the points in S corresponding to the respective Voronoi cells of x_1 and x_2.

First, consider the case where $\text{sign}(f(p_1)) \neq \text{sign}(f(p_2))$, i.e., x_1 and x_2 are on opposite sides of the boundary, B. In this case,

$$|g(x_1) - g(x_2)| = \frac{(d(x_1) + d(x_2))}{\epsilon},$$

and thus it suffices to show that $d(x_1) + d(x_2) \leqslant ||x_1 - x_2||$.

Assume for the sake of contradiction that $d(x_1) + d(x_2) > ||x_1 - x_2||$. Note that because x_1 and x_2 belong in Voronoi cells with different classifications from f, the line segment connecting x_1 and x_2 must cross the boundary, B, at some point c. Therefore, $||x_1 - c|| + ||x_2 - c|| = ||x_1 - x_2|| < d(x_1) + d(x_2)$; without loss of generality, this implies that $||x_1 - c|| < d(x_1)$. But since $c \in F \in B$, this contradicts that $d(x_1)$ is the minimum distance from x_1 to B.↯

Next, consider the case where $\text{sign}(f(p_1)) = \text{sign}(f(p_2))$. In this case

$$|g(x_1) - g(x_2)| = \frac{|d(x_1) - d(x_2)|}{\epsilon},$$

and thus, without loss of generality, it suffices to show that $d(x_1) - d(x_2) \leqslant ||x_1 - x_2||$.

Assume for the sake of contradiction that $d(x_1) - d(x_2) > ||x_1 - x_2||$. Thus $d(x_1) > ||x_1 - x_2|| + d(x_2)$. However, this suggests that we can take the path from x_1 to x_2 to B with a smaller total distance than $d(x_1)$, contradicting that $d(x_1)$ is the minimum distance from x_1 to B.↯

We now show that $\forall p \in S$, $|g(p)| \geqslant 1$, i.e., $d(p) \geqslant \epsilon$. In other words, we must show that the distance of any point, $p \in S$ to the boundary, B, is at least ϵ. Consider a point, x, on some face, $F_{ij} \in B$. This point is equidistant from p_i and $p_j \in S$, on which f makes different predictions; and every other point in S is at least as far from x as p_i and p_j. I.e., $||p_i - x|| = ||p_j - x|| \leqslant ||p - x||$, $\forall p \in S$. By the triangle inequality, $2||p_i - x|| \geqslant ||p_i - p_j||$, and by Lemma 6.4, $||p_i - p_j|| \geqslant 2\epsilon$. Thus $||p - x|| \geqslant \epsilon$, $\forall p \in S$; therefore every point on the boundary is at least ϵ from $p \in S$.

Putting everything together, we have that $\forall p \in S$, $|g(p)| \geqslant 1 \geqslant \epsilon K_G$. □

method	*clean (%)*	*PGD (%)*	*VRA(%)*
MNIST ($\epsilon = 0.3$)			
ReLU GloRo	**98.7**	97.4	**94.6**
BCP	93.4	89.5	84.7
KW	98.9	**97.8**	94.0
MMR	98.2	96.2	93.6
MNIST ($\epsilon = 1.58$)			
ReLU GloRo	**92.8**	67.0	**51.9**
LMT	86.5	53.6	40.6
BCP	92.4	65.8	47.9
KW	88.1	**67.9**	44.5
CIFAR-10 ($\epsilon = 36/255$)			
ReLU GloRo	**67.9**	**61.3**	51.0
LMT	63.1	58.3	38.1
BCP	65.7	60.8	**51.3**
KW	60.1	56.2	50.9

Table A.2: Comparison of ReLU-based GloRo Nets against wide set of certifiable training approaches approaches that also use ReLU-based architectures, showing the VRA, PGD accuracy, and clean accuracy of each approach. Best results are highlighted in bold.

Note that while Theorem 6.3 is stated for binary classifiers, the result holds for categorical classifiers as well. We can modify the construction of g from the above proof in a straightforward way to accommodate categorical classifiers. In the case where there are m different classes, the output of g has m dimensions, each corresponding to a different class. Then, for x in a Voronoi cell corresponding to $p_x \in S$ with label, j, we define $g_j(x) ::= d(x)/\epsilon$ and $g_i(x) ::= 0 \ \forall i \neq j$. We can see that, for all pairs of classes, i and j, the Lipschitz constant of $g_i - g_j$ in this construction is the same as the Lipschitz constant of g in the above proof, since only one dimension of the output of g changes at once. Thus, we can use the global bound suggested in Appendix A.1 to certify the points in S.

A.3 Results on ReLU Networks

Here we include additional results showing that GloRo Nets trained with ReLU activations outperform other methods that use ReLU-based architectures. The results are shown in Table A.2.

A.4 Hyperparameters

The full set of hyperparameters used for the experiments in Section 6.4 are shown in Table A.3, and those for the additional experiments in Appendix A.3 are shown in Table A.4 We explain each column below and discuss how a particular value is selected for each hyperparameter. Code for reproducing our results can be found at `https://github.com/klasleino/gloro`.

Architectures. To denote architectures, we use $c(C, K, S)$ to denote a convolutional layer with C output channels, a kernel of size $K \times K$, and strides of width S. We use `SAME` padding

architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	MNIST	`None`	0	512	500	0.3	0.3
2C2F GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`orthogonal`	1e-3	`decay_to_1e-6`	`0.1,2.0,500`	`single`	5	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	MNIST	`None`	0	512	200	1.74	1.58
4C3F GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`orthogonal`	1e-3	`decay_to_1e-6`	`1.5`	`log`	5	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	CIFAR-10	`tfds`	0	512	800	0.141	0.141
6C2F GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`orthogonal`	1e-3	`decay_to_1e-6`	`1.2`	`log`	5	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	Tiny-Imagenet	`default`	0	512	800	0.141	0.141
8C2F GloRo	*optimizer*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`default`	1e-4	`decay_to_5e-6`	`1.2,10,800`	`log`	5	

Table A.3: Hyperparameters used for training (MinMax) GloRo Nets.

architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	MNIST	`None`	0	256	500	0.45	0.3
2C2F							
GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`default`	1e-3	`decay_to_1e-6`	`0,2,500`	`single`	10	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	MNIST	`None`	0	256	300	1.75	1.58
4C3F							
GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`default`	1e-3	`decay_to_5e-6`	`0,3,300`	`single`	10	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	CIFAR-10	`default`	20	256	800	0.1551	0.141
6C2F							
GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`default`	1e-3	`decay_to_1e-6`	`1.2`	`log`	5	
architecture	*dataset*	*data augmentation*	*warm-up*	*batch size*	*# epochs*	ϵ_{train}	ϵ_{test}
	Tiny-Imagenet	`default`	0	256	500	0.16	0.141
8C2F							
GloRo	*initialization*	*init_lr*	*lr_decay*	*loss*	*ϵ schedule*	*power_iter*	
	`default`	2.5e-4	`decay_to_5e-7`	`1,10,500`	`single`	1	

Table A.4: Hyperparameters used for training (ReLU) GloRo Nets.

unless noted otherwise. We use $d(D)$ to denote a dense layer with D output dimensions. We use MinMax [4] or ReLU activations (see Appendix A.3) after each layer except the top of the network, and do not include an explicit Softmax activation. Using this notation, the architectures referenced in Section 6.4 are as shown in the following list.

- 2C2F: c(16,4,2).c(32,4,2).d(100).d(10)
- 4C3F: c(32,3,1).c(32,4,2).c(64,3,1).c(64,4,2).d(512).d(512).d(10)
- 6C2F: c(32,3,1).c(32,3,1).(32,4,2).(64,3,1) .c(64,3,1).c(64,4,2).d(512).d(10)
- 8C2F: c(64,3,1).c(64,3,1).c(64,4,2).c(128,3,1).c(128,4,2).c(256,3,1).(256,4,2).d(200)

We arrived at these architectures in the following way. 2C2F, 4C3F and 6C2F are used in prior work [30, 82, 157] to evaluate the verifiable robustness, and we used them to facilitate a direct comparison on training cost and verified accuracy. For Tiny-ImageNet, we additionally explored the architecture described in [82] for use with that dataset, but found that removing one dense and one convolutional layer (denoted by 8C2F in the list above) produced the same (or better) verified accuracy, but lowered the total training cost.

Data Preprocessing. For all datasets, we scaled the features to the range [0,1]. On some datasets, we used the following data augmentation pipeline `ImageDataGenerator` from `tf.keras`, which is denoted by `default` in Table A.4 and A.3.

```
rotation_range=20
width_shift_range=0.2
height_shift_range=0.2
horizontal_flip=True
shear_range=0.1
zoom_range=0.1
```

When integrating our code with `tensorflow-dataset`, we use the following augmentation pipeline and denote it as `tfds` in Table A.4 and A.3.

```
horizontal_flip=True
zoom_range=0.25
random_brightness=0.2
```

Our use of augmentation follows the convention established in prior work [82, 157]: we only use it on CIFAR and Tiny-Imagenet, but not on MNIST.

Epsilon Scheduling. Prior work has also established a convention of gradually scaling ϵ up to a value that is potentially larger than the one used to measure verified accuracy leads to better results. We made use of the following schemes for accomplishing this.

- No scheduling: we use '`single`' to denote we ϵ_{train} for all epochs.
- Linear scheduling: we use a string '`x,y,e`' to denote the strategy that at training epoch t, we use $\epsilon_t = x + (y - x) * (t/e)$ if $t \leqslant e$. When $t > e$, we use the provided ϵ_{train} to keep training the model.

- Logarithmic scheduling: we use '`log`' to denote that we increase the epsilon with a logarithmic rate from 0 to ϵ_{train}.

We found that scheduling ϵ is often unnecessary when instead scheduling the TRADES parameter λ (discussed later in this section), which appears to be more effective for that loss. To select a scheme for scheduling ϵ, we compared the results of the three options listed above, and selected the one that achieved the highest verified accuracy. If there was no significant difference in this metric, then we instead selected the schedule with the least complexity, assuming the following order: `single`, (`x,y,e`), `log`. When applying (`x,y,e`) and `log`, we began the schedule on the first epoch, and ended it on (# epochs)/2.

Initialization & Optimization. In Table A.4, `default` refers to the Glorot uniform initialization, given by `tf.keras.initializers.GlorotUniform()`. The string '`ortho`' refers to an orthogonal initialization given by `tf.keras.initializers.Orthogonal()`. To select an initialization, we compared the verified accuracy achieved by either, and selected the one with the highest metric. In the case of a tie, we defaulted to the Glorot uniform initialization. We used the `adam` optimizer to perform gradient descent in all experiments, with the initial learning rate specified in Table A.4 and A.3, and default values for the other hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-07$, `amsgrad` disabled).

Learning Rate Scheduling. We write '`decay_to_lb`' to denote a schedule that continuously decays the learning rate to `lb` at a negative-exponential rate, starting the decay at (#epochs)/2. To select `lb`, we searched over values `lb` $\in \{1\times10^{-7}, 5\times10^{-7}, 1\times10^{-6}, 5\times10^{-6}\}$, selecting the value that led to the best VRA. We note that for all datasets except Tiny-Imagenet, we used the default initial rate of 1×10^{-3}. On Tiny-Imagenet, we observed that after several epochs at this rate, as well as at 5×10^{-4}, the loss failed to decrease, so again halved it to arrive at 2.5×10^{-4}.

Batch size & Epochs. For all experiments, we used mini-batches of 256 instances. Because our method does not impose a significant memory overhead, we found that this batch size made effective use of available hardware resources, increasing training time without impacting verified accuracy, when compared to minibatch sizes 128 and 512. Because the learning rate, ϵ, and λ schedules are all based on the total number of epochs, and can have a significant effect on the verified accuracy, we did not monitor convergence to determine when to stop training. Instead, we trained for epochs in the range $[100, 1000]$ in increments of 100, and when verified accuracy failed to increase with more epochs, attempted training with fewer epochs (in increments of 50), stopping the search when the verified accuracy began to decrease again.

Warm-up. Lee et al. [82] noted improved performance when models were pre-trained for a small number of epochs before optimizing the robust loss. We found that this helped in some cases with GloRo networks as well, in particular on CIFAR-10, where we used the same number of warm-up epochs as prior work.

TRADES Scheduling. When using the TRADES loss described in Section 6.3, we found that scheduling λ often yielded superior results. We use '`x,y,e`' to denote that at epoch t, we set $\lambda_t = x + (y - x) * (t/e)$ if $t < e$ else $\lambda_t = y$. We write '`x`' to denote we use $\lambda = x$ all the time. To select the final λ, we trained on values in the range $[1, 10]$ in increments of 1,

and on finding the whole number that yielded the best result, attempted to further refine it by increasing in increments 0.1.

Power Iteration. As discussed in Section 6.3, we use power iteration to compute the spectral norm for each layer to find the layer-wise Lipschitz constants. In Table A.4, *power_iter* denotes the number of iterations we run for each update during training. We tried values in the set $\{1, 2, 5, 10\}$, breaking ties to favor fewer iterations for less training cost. After each epoch, we ran the power iteration until convergence (with tolerance 1×10^{-5}), and all of the verified accuracy results reported in Section 6.4 are calculated using a global bound based on running power iteration to convergence as well. Since the random variables used in the power iterations are initialized as `tf.Variables`, they are stored in `.h5` files together with the architecture and network parameters. Therefore, one can directly use the converged random variables from the training phase during the test phase.

Search Strategy. Because of the number of hyperparameters involved in our evaluation, and limited hardware resources, we did not perform a global grid search over all combinations of hyperparameters discussed here. We plan to do so in future work, as it is possible that results could improve as we may have missed better settings than those explored to produce the numbers reported in our evaluation. Instead, we adopted a greedy strategy, prioritizing the choices that we believed would have the greatest impact on verified accuracy and training cost. In general, we explored parameter choices in the following order: ϵ schedule, λ schedule, # epochs, LR decay, warm-up, initialization, # power iterations, mini-batch size.

A.5 Measuring Memory Usage

In our experiments, we used TensorFlow to train and evaluate standard and GloRo networks, and Pytorch to train and evaluate KW and BCP (since Wong et al. [157] and Lee et al. [82] implement their respective methods in Pytorch). To measure memory usage, we invoked `tf.contrib.memory_stats.MaxBytesInUse()` at the end of each epoch for standard and GloRo networks, and took the peak active use from `torch.cuda.memory_summary()` at the end of each epoch for KW and BCP.

We note that some differences may arise as a result of differences in memory efficiency between Tensorflow and Pytorch. In particular, Pytorch enables more control over memory management than does Tensorflow. In order to mitigate this difference as much as possible, we did not disable gradient tracking when evaluating certification times and memory usage in Pytorch. While gradient tracking is unnecessary for certification (it is only required for training), Tensorflow does not allow this optimization, so by forgoing it the performance results recorded in Figure 6.8b in Section 6.4 are more comparable across frameworks.

Appendix B

Further Details on Robustness Relaxations

B.1 Discussion of the Method Proposed by Jia et al.

In recent work, Jia et al. [68] also proposed an approach that aims to capture certified robustness for top-k predictions. In this section, we provide details on differences between this work and ours; particularly the shortcomings of this approach that our work addresses.

Recall our notation from Section 7.2, where, for neural network, f, we define $F^k(x)$ as the set of classes corresponding to the top k logit values in $f(x)$. The approach of Jia et al. provides a *probabilistic* bound on the radius, $r_j^k(x)$, under which a given class $j \in F^k(x)$ will remain in the top-k predictions. That is, Jia et al. provide a probabilistic guarantee that Equation B.1 holds.

$$\forall x' \ . \ ||x - x'|| \leqslant r_j^k(x) \implies j \in F^k(x') \tag{B.1}$$

In their evaluation, Jia et al. consider a point, x, to be *certified* at radius, ϵ, if $r_{y^*}^k \geqslant \epsilon$, where y^* is the *ground truth* label for x. This presents a problem for certifying unseen points as the ground truth cannot be known. We therefore stipulate that certification must be independent of the true label of the point being certified. Moreover, replacing the ground truth with the predicted label is unsatisfactory, because the purpose of generalizing to top-k predictions is to consider cases where *any* of the predictions in $F^k(x)$ may be correct.

While Jia et al. do not address this issue, one straightforward adaptation of their approach is to take the minimum certified radius over all classes in $F^k(x)$. That is, let $r^k(x) = \min_{j \in F^k(x)}\{r_j^k(x)\}$. We see that this leads to a natural guarantee given by Equation B.2, which can be certified without knowledge of the ground truth class. In short, Equation B.2 holds because by taking the minimum r_j^k, we are guaranteed that *all* of the top k classes will remain in the top k classes under perturbations bounded by $r^k(x)$.

$$\forall x' \ . \ ||x - x'|| \leqslant r^k(x) \implies F^k(x) = F^k(x') \tag{B.2}$$

We note however, that when certification is determined by comparing $r^k(x)$ to a fixed radius, Equation B.2 is equivalent to our proposed definition for *top-k robustness* (Definition 7.1). As discussed in Section 7.2, this is also *not* a satisfactory robustness analogue to

top-k accuracy as it does not relax local robustness. We therefore argue that *RTK robustness* (Definition 7.2) should be used to certify the robustness of top-k predictions.

B.2 Correctness of RTK GloRo Nets

Theorem. *Let g be an RTK GloRo Net as defined by Equation 7.1. Then, if the maximal output of $g(x)$ is not $\bot$, then F is RTK ϵ-locally-robust at x.*

Proof. Let $y = F(x) = \text{argmax}_i\{f_i(x)\}$. Assume that the maximal output of $g(x)$ is not $\bot$, i.e., $\exists i$ such that $g_\bot(x) < g_i(x) \leqslant g_y(x) = f_y(x)$. By the definition of $g_\bot$ in Equation 7.1, we obtain (B.3). By the definition of y, we obtain (B.4).

$$f_y(x) > \max_i\{f_i(x) - m(x)\} \tag{B.3}$$

$$= f_y(x) - m(x) \tag{B.4}$$

Thus, we have that $m(x)$ is positive. As $m(x)$ is defined as $\max_{k \leqslant K}\{m^k(x)\}$, this means that there exists some $k^* \leqslant K$ such that $m^{k^*}(x) > 0$ (B.5).

We recall from the definition of RTK robustness, we must show that there exists some $k \leqslant K$ such that for all x' at distance no greater than ϵ from x, $F^k(x) = F^k(x')$. We proceed to show that k^* is such a k; i.e., $||x - x'|| \leqslant \epsilon \implies F^{k^*}(x) = F^{k^*}(x')$.

From (B.5) we expand the definition of $m^k(x)$ to obtain (B.6); and the definition of m^k_j to obtain (B.7).

$$0 < m^{k^*}(x) \tag{B.5}$$

$$= \min_{j \in F^{k^*}(x)} \left\{ m^{k^*}_j(x) \right\} \tag{B.6}$$

$$= \min_{j \in F^{k^*}(x),\ i \notin F^{k^*}(x)} \left\{ f_j(x) - f_i(x) - \epsilon \mathcal{K}_{ji} \right\} \tag{B.7}$$

We observe that (B.7) implies (B.8).

$$\forall j \in F^{k^*}(x),\ i \notin F^{k^*}(x) \ \ . \ \ f_i(x) + \epsilon \mathcal{K}_{ji} < f_j(x) \tag{B.8}$$

Next, we assume x' satisfies $||x - x'|| \leqslant \epsilon$. As $\mathcal{K}_{ji}$ is an upper bound on the Lipschitz constant of $f_j - f_i$ we obtain (B.9).

$$\begin{aligned} & \frac{|f_j(x) - f_i(x) - (f_j(x') - f_i(x'))|}{||x - x'||} \leqslant \mathcal{K}_{ji} \\ \implies \quad & |f_j(x) - f_i(x) - (f_j(x') - f_i(x'))| \leqslant \mathcal{K}_{ji}\epsilon \end{aligned} \tag{B.9}$$

Thus we argue as follows for all $j \in F^{k^*}(x)$ and $i \notin F^{k^*}(x)$. First, by applying (B.9), we obtain (B.10). Then, by applying (B.8) we obtain (B.11).

$$\begin{aligned} & f_i(x) + f_j(x) - f_i(x) - f_j(x') + f_i(x') \\ \leqslant\ & f_i(x) + |f_j(x) - f_i(x) - f_j(x') + f_i(x')| \\ \leqslant\ & f_i(x) + \epsilon \mathcal{K}_{ji} && \text{(B.10)} \\ <\ & f_j(x) && \text{(B.11)} \end{aligned}$$

By rearranging the terms in the above inequality, we obtain (B.12).

$$\forall j \in F^{k^*}(x),\ i \notin F^{k^*}(x)\ \ .\ \ f_i(x') < f_j(x') \tag{B.12}$$

Finally, we realize that (B.12) is equivalent to $F^{k^*}(x) = F^{k^*}(x')$. To see why, consider the following. Let $Z = \{\forall i\ .\ f_i(x')\}$ be the set of all logit values produced by f on x' (assume WLOG each logit value is unique), and let $Z_{k*} = \{\forall j \in F^{k^*}(x)\ .\ f_j(x')\}$ be the subset of Z containing the logit values of $f(x')$ corresponding to the classes in $F^{k^*}(x)$. By (B.12) we have that for all $z \in Z_{k*}$ and $z' \in Z \backslash Z_{k*}$, $z > z'$. Thus, Z_{k*} contains the top k^* elements of Z.

Putting everything together, we conclude that $\exists k \leqslant K\ :\ ||x - x'|| \leqslant \epsilon \implies F^k(x) = F^k(x')$. □

B.3 Discussion of Other Certification Techniques

While many certification methods have been proposed and studied in the past, we use the GloRo Net approach to certification for two key reasons. First, GloRo Nets have been shown to be among the top state-of-the-art methods for deterministic ℓ_2 certification, matching or outperforming the VRA of all other deterministic methods recently surveyed by Leino et al. [88]. Second, GloRo Nets provide an elegant way for incorporating our robustness objectives into a *certifiable training procedure*, as certified training can be reduced to simply performing overapproximate certification.

However, leveraging other types of techniques for certifying our proposed robustness properties remains an interesting possibility for future work to explore. We now present a discussion of how other certification techniques in the literature may be adapted to incorporate RTK and affinity robustness.

Randomized Smoothing. Jia et al. [68] suggest a technique based on randomized smoothing that is in the same spirit as RTK robustness. While we point out several problems with their approach, Appendix B.1 describes how to adapt their technique to get an equivalent property to top-k robustness (Definition 7.1). However, as we argue in Section 7.2, RTK robustness (Definition 7.2) is the correct analogue to top-k accuracy.

In general, RTK robustness can be certified given a method to certify top-k accuracy, by iteratively checking whether top-k robustness holds for any k in $\{1, ..., K\}$. Thus, it should be possible to adapt the method of Jia et al. to correctly certify RTK robustness via randomized smoothing. A naive implementation of this would be very expensive (following the approach outlined in Appendix B.1), however, as it would require $O(K^2)$ calls to Jia et al.'s method, which is already expensive. On hardware similar to our own, we estimate such an implementation would take about a minute to evaluate and certify each instance.[1] Future work may be able to determine a more efficient way to achieve this.

Finally, we note that although randomized smoothing can, in theory, obtain certificates without any requirements on the original model, in order to get reasonable performance, it is necessary to augment the data with Gaussian noise during training—this allows the model to behave well under the noise that will be introduced at evaluation time. It is therefore possible that for similar reasons, additional training considerations would be important to make use of the relaxed guarantee in smoothed models.

[1] This computation would require evaluating the underlying model on 100,000 samples (the number of samples required for adequate smoothing [26]) K^2 times.

1-Lipschitz Models. Recently, a few similar certification approaches to GloRo Nets have been proposed, which use orthonormal projections to ensure the model is 1-Lipschitz [147]. When the model is 1-Lipschitz, certification simply requires a margin of ϵ between the top logit output and the others. Because these techniques achieve robustness through Lipschitzness, the certification procedure in Algorithm 7.1 would also apply to such methods.

Primarily, these approaches differ from GloRo Nets in the way they are *trained*, targeting robustness though a hinge-like loss function that encourages a sufficiently large margin between classes. Because of the nuances of RTK robustness, it is less clear how RTK robustness could be achieved with a simple loss function (e.g., RTK robustness is a naturally disjunctive property, and the network should have a choice in which k to target). However, the GloRo training we propose avoids the challenges of redesigning a proper loss function, and would still work to train models with orthonormalized kernels.

Convex Relaxation/Bound Propagation. It is not immediately clear to us how to achieve RTK (or Affinity) robustness with approaches like KW [157] or IBP-based methods [82], but we believe this could be an interesting future direction. It may be possible to change these methods to ensure a margin between the k^{th} class and the $(k+1)^{\text{th}}$ class. This in turn could be used as a step in certifying RTK robustness as noted in our discussion of Randomized Smoothing. Note, however, that this would likely mean that KW and IBP would require multiple propagations to handle RTK robustness, making them more expensive (while the analysis of the GloRo approach only needs to consider the logits, which only need to be computed once).

An additional important issue, moreover, is incorporating RTK robustness into the learning objective. Because of the disjunctive nature of RTK robustness, it is not obvious how to design the loss function to promote it. GloRo Nets avoid this issue because they essentially make an equivalence between the problem of certification and the robust learning objective.

B.4 Correctness of Affinity GloRo Nets

Theorem. *Let g be an Affinity GloRo Net as defined by Equation 7.3. Then, if the maximal output of $g(x)$ is not $\bot$, then F is affinity ϵ-locally-robust at x.*

Proof. The proof follows a similar approach to the proof of Theorem 7.1. Let $y = F(x) = \text{argmax}_i\{f_i(x)\}$. Assume that the maximal output of $g(x)$ is not $\bot$, i.e., $\exists i$ such that $g_\bot(x) < g_i(x) \leqslant g_y(x) = f_y(x)$. By the definition of $g_\bot$ in Equation 7.3, we obtain (B.13). By the definition of y, we obtain (B.14).

$$f_y(x) > \max_i\{f_i(x) - m(\mathcal{S}, x)\} \tag{B.13}$$

$$= f_y(x) - m(\mathcal{S}, x) \tag{B.14}$$

Thus, we have that $m(\mathcal{S}, x)$ is positive. From the definition of $m(\mathcal{S}, x)$ in Equation 7.2, we conclude that there exists some k^* and some $S^* \in \mathcal{S}$, such that $F^{k^*}(x) \subseteq S^*$ and $m^{k^*}(x) > 0$.

We recall from the definition of affinity robustness, we must show that there exists some k and S such that for all x' at distance no greater than ϵ from x, $F^k(x) = F^k(x')$ and $F^k(x) \subseteq S$. We proceed to show that $||x - x'|| \leqslant \epsilon \implies F^{k^*}(x) = F^{k^*}(x') \wedge F^{k^*}(x) \subseteq S^*$.

From our observations above, we have that $F^{k^*}(x) \subseteq S^*$, therefore it suffices to simply show that $||x - x'|| \leqslant \epsilon \implies F^{k^*}(x) = F^{k^*}(x')$, given that $m^{k^*}(x) > 0$. As $m^k(x)$ is defined the same for both Affinity GloRo Nets (Equation 7.3) and RTK GloRo Nets (Equation 7.1), the remainder of the proof proceeds exactly as the proof for Theorem 7.1 in Appendix B.2. □

B.5 Experimental Details and Hyperparameters

Hardware. All experiments were run on an NVIDIA TITAN RTX GPU with 24 GB of RAM, and a 4.2GHz Intel Core i7-7700K with 32 GB of RAM.

Data Splits. On CIFAR-100 and Tiny-Imagenet, we used the standard train-test split. We are unaware of any "standard" train-test split for EuroSAT; thus we used 2/3 of the data for the training set and 1/3 of the data for the test set.

Architectures. For CIFAR-100 and EuroSAT we used a simple convolutional architecture consisting of two blocks with width 32 and 64 respectively—each containing a convolutional layer with 3×3 followed by a down-sampling layer—followed by two dense hidden layers with width 256 each. For Tiny-Imagenet we used the same architecture as was used previously by Lee et al. [82] and subsequently Leino et al. [88]. This architecture consists of three convolutional blocks followed by a dense layer of width 256. The first block is composed of two convolutional layers with 3×3 filters and 64 channels followed by a strided convolution with 4×4 filters and 64 channels. The first block is the same as the first block, but with a width of 128. The third block has one convolutional layer with 3×3 filters and 256 channels followed by a strided convolution with 4×4 filters and 256 channels. For ACAS Xu, we used a dense network consisting of three hidden layers with 1,000 neurons each. In all networks, we used *min-max* activations [4] rather than ReLU activations, as these have been observed to achieve better performance with GloRo Nets [88]. Similarly, down-sampling in the CIFAR-100 and EuroSAT models was achieved via *invertible down-sampling* [4] rather than max-pooling.

Hyperparameters. Details on the hyperparameters used to train each model presented in the evaluation in Section 7.4 are given in Table B.1. All models were trained using the ADAM optimizer.

GloRo Nets (and our variations thereof in Chapter 7) make use of the underlying model's Lipschitz constant, which is approximated using the power method (see [88] for more details). Prior to evaluation, the power method is run to convergence, however, during training we run a fixed number of iterations on each batch. For each of the models in our evaluation we used two power iterations on each training batch.

We used a continuous learning rate schedule, where the learning rate was adjusted on each epoch. A description of the learning rate schedule is given in Table B.1. E.g., learning rate schedule of "$10^{-3} \to 10^{-6}$ after half" means that the learning rate begins at 10^{-3} and halfway through training the learning rate decays exponentially to reach 10^{-6} on the final epoch.

For each model, we used one of two loss functions adapted for GloRo Nets as proposed by Leino et al. [88], specified in the "loss function" column of Table B.1: *cross-entropy* or *TRADES* (see [88] for more details on these loss functions). The TRADES loss function takes

dataset	*guarantee*	*epochs*	*batch size*	*learning rate schedule*	*loss function*	*TRADES schedule*
EuroSAT	standard	200	256	$10^{-3} \to 10^{-6}$ after half	TRADES	$0.01 \to 1.2$ log half
EuroSAT	RT3	200	256	$10^{-3} \to 10^{-6}$ after half	TRADES	$1.0 \to 1.2$
EuroSAT	highway+river affinity	200	256	$10^{-3} \to 10^{-6}$ after half	cross-entropy	N/A
EuroSAT	highway+river+agriculture affty.	200	256	$10^{-3} \to 10^{-6}$ after half	cross-entropy	N/A
CIFAR-100	standard	200	256	$10^{-3} \to 10^{-6}$ after half	TRADES	$0.01 \to 1.2$ log half
CIFAR-100	RT5	200	256	$10^{-3} \to 10^{-6}$ after half	TRADES	$0.01 \to 1.2$ log half
CIFAR-100	superclass affinity	200	256	$10^{-3} \to 10^{-6}$ after half	TRADES	$0.01 \to 1.2$ log half
Tiny-Imagenet	RT5	200	256	$2.5 \cdot 10^{-5} \to 5 \cdot 10^{-6}$ after half	TRADES	$1.0 \to 10.0$ half
ACAS Xu	targeted affinity	100	128	$10^{-3} \to 5 \cdot 10^{-6}$ after half	cross-entropy	N/A

Table B.1: Hyperparameters for each model used in the evaluation in Section 7.4.

an additional hyperparameter, λ, which was scheduled during training. Where applicable, the "*TRADES schedule*" column describes how this parameter was scheduled over training. For example, "$0.01 \to 1.2$ log half" means that λ was initialized to 0.1 and was increased logarithmically each epoch (i.e., it increases at a decreasing rate) to reach a final value of 1.2 halfway through training; and "$1.0 \to 1.2$" means that λ was initialized to 1.0 and was increased linearly to reach 1.2 on the final epoch.

B.6 Further CIFAR-100 Results

Additional Samples of RT5-Certifiable Points. In Section 7.4.3, we provide a few selected samples of correctly-classified, RT5-certifiable points with their corresponding robust prediction sets (Figure 7.6). Figure B.2 provides a larger random sample of such examples, illustrating the types of non-top-1 guarantees that are typically achieved by an RT5 GloRo Net on CIFAR-100.

Comparing Top-k Relaxations to Reducing the Number of Classes. To an extent, relaxations like RTK robustness are somewhat similar to considering a classification problem with fewer unique classes. We note, however, that achieving RTK VRA is still more difficult than achieving the same standard VRA with $1/K$ the number of classes. This is because the loosest top-k guarantee on a given point might be some $k < K$, in which case the correct class for that instance would need to be among the top k classes rather than the top K (see Section 7.4.1:**Metrics** for details on how RTK and standard VRA are computed). E.g., some points receive only a top-1 guarantee even under a relaxed robustness variant, meaning that for those points to count towards the RTK VRA, they would need to be classified *exactly correctly*, not simply result in the correct label appearing among the top K outputs.

For the sake of examining this juxtaposition, we trained a network on CIFAR-100 for RT10 robustness in order to compare against state-of-the-art results on CIFAR-10 (which has similar images and $1/K$ the number of classes). Leino et al. [88] find the state-of-the-art standard VRA on CIFAR-10 with radius of $\epsilon = 0.141$ to be approximately 58%. Meanwhile, we were able to achieve approximately 55% RT10 VRA on CIFAR-100 with the same radius. Thus, these findings are essentially in line with the intuition that RTK robustness is similar to, but slightly more challenging than the objective of standard robustness with $1/K$ the number of classes.

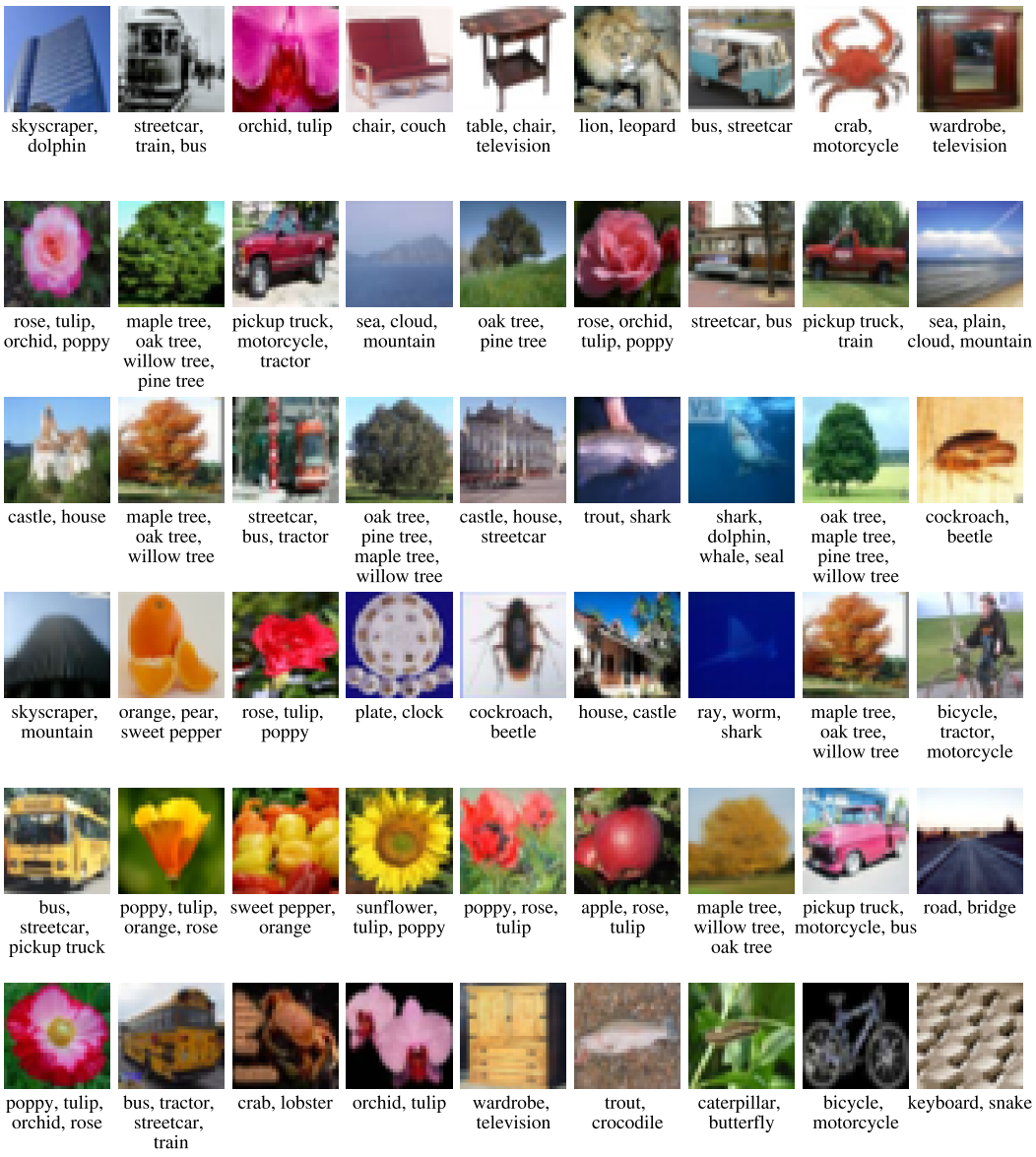


Figure B.2: Random samples of CIFAR-100 instances that are both correctly classified and certified as top-k robust for $k > 1$. The classes included in the RT5 robustness guarantee are given beneath each image.

Appendix C

Further Details on White-box Membership Inference

C.1 Defenses Against Membership Inference

Concerns about privacy, underscored by concrete threats such as the white-box membership inference attacks developed in Chapter 8, have also motivated research to provide adequate defenses against such threats. In this section we explore the ability of some of the commonly-proposed mitigation techniques to defend against our attack. In particular, we focus on *differential privacy* [34] and regularization. We find that, while both are useful to a degree, neither dropout nor ϵ-differentially private training with a large ϵ, are necessarily sufficient for mitigating the privacy risk posed by our attack.

Differential Privacy

Differential privacy (DP) [34] is often seen as the gold standard for private models, as models trained with differential privacy have provable guarantees against membership inference. Namely, Yeom et al. [161] showed that, given an ϵ-differentially private learning algorithm, an adversary can achieve an advantage of at most $e^{\epsilon}-1$. Differential privacy has been applied to many areas of machine learning, including logistic regression [22], SVMs [120], and more recently, deep learning [1, 127]. However, current methods for ensuring differential privacy are typically costly with respect to the accuracy of the model, particularly for small values of ϵ, which give a better privacy guarantee. For this reason, in practice, ϵ is often chosen to be quite large; for example, in 2017, Apple was found to use an effective epsilon as high as 16 in some of its routines [142].

We used the Tensorflow Privacy library [100], an implementation of the *moments accountant* method [1], which guarantees (ϵ, δ)-differential privacy, to study the practical efficacy of our attack on protected models. This method utilizes several hyperparameters from which ϵ is derived; for uniformity, we modified only the *noise multiplier* to achieve the desired ϵ, and used heuristics described in the original paper [1] to select the remaining hyperparameters. While a different tuning of the hyperparameters may result in a different privacy-utility trade-off, the privacy guarantee depends only on ϵ and δ, *not the hyperparameters directly.* In each case, δ was selected to be smaller than $1/N$ where N is the size of the dataset.

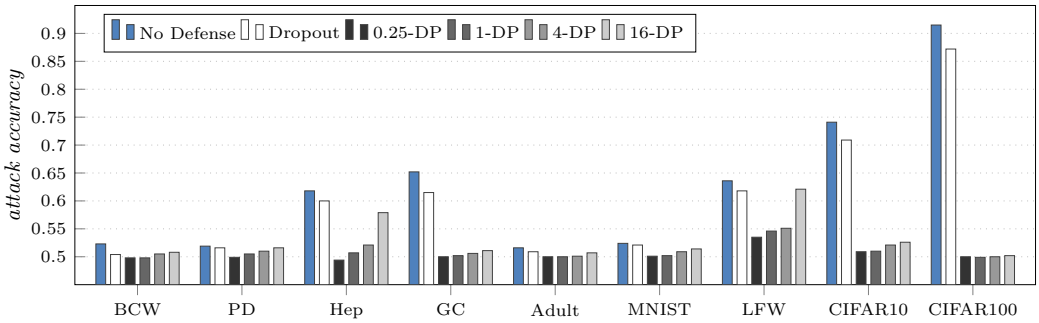


Figure C.1: Attack accuracies against models trained with either dropout or (ϵ, δ)-differential privacy for various values of ϵ.

Figure C.1 shows the effectiveness the nn-wb attack against models trained with differential privacy for various values of ϵ on each dataset. The train and test accuracies of the corresponding differentially-private target models are shown in Table C.2. First, we note that as expected, when ϵ decreases the adversary's effectiveness quickly declines. However, when ϵ is large ($\epsilon = 16$), our attack occasionally performs *essentially the same* on the differentially-private model as on the undefended model. For example, on BCW, PD, and LFW, 16-DP provided less defense than simple regularization, while harming the accuracy of the model. Similarly, on Hep, 16-DP reduced the effectiveness of nn-wb, but not below the effectiveness of shadow-bb on the corresponding undefended model. These findings suggest that the practical benefits of large-ϵ-differential privacy cannot be taken for granted; in general, differential privacy may only be effective for sufficiently small ϵ.

Nevertheless, it is clear that a practical adversary is unlikely to achieve performance that is tight with the theoretical bound. For both the undefended model and the models trained with DP for $\epsilon > \ln 2 \approx 0.69$, the theoretical bound on the adversary's accuracy is 100%, which no attack was able to achieve. On the other hand, for $\epsilon = 0.25$, the theoretical maximum accuracy of the adversary is 64.2%. In most such cases, our attack fared far poorer than this, coming closest on LFW, where our attack achieved 53.5% accuracy (25% of the theoretical maximum advantage) on the 0.25-DP model. Thus, we conclude that because the accuracy of a real adversary is not likely to be tight with the worst-case guarantee, it is indeed pragmatic to select a somewhat large ϵ. However, our evaluation shows that ϵ should not be chosen to be too large, or else the operative benefits of differential privacy may be lost. Furthermore, the success of a given value of ϵ appears to vary across different datasets and models. One must therefore be careful when making a practical selection for ϵ; to this end, we suggest that our attack may be useful in assessing which values of ϵ are appropriate for a given application.

An apparent drawback of the examined method for obtaining differential privacy, revealed in our evaluation, is the steep cost in performance (Figure C.2), which is particularly high for small ϵ. Despite the fact that our attack became far less effective for small ϵ, this cost limits the practicality of the defense, highlighting the need for more research in this area. The results we find here align with recent work [67], in which Jayaraman and Evans showed that the privacy leakage tends to increase as ϵ becomes large enough to avoid a significant loss in accuracy. Indeed, only on the German Credit dataset did 16-DP provide a good defense while nearly maintaining the accuracy of the unprotected model. In the other cases we evaluated, either our attack performed comparably on the DP and unprotected models,

dataset		*no defense*	*dropout*	$\epsilon = 0.25$	$\epsilon = 1$	$\epsilon = 4$	$\epsilon = 16$
BCW	*train*	0.987	0.982	0.601	0.654	0.767	0.778
	test	0.944	0.961	0.609	0.675	0.763	0.808
PD	*train*	0.789	0.784	0.680	0.678	0.681	0.683
	test	0.756	0.783	0.673	0.651	0.649	0.654
Hep	*train*	0.997	0.992	0.534	0.695	0.700	0.729
	test	0.810	0.849	0.555	0.786	0.803	0.817
GC	*train*	0.937	0.932	0.625	0.656	0.680	0.707
	test	0.701	0.730	0.610	0.661	0.687	0.698
Adult	*train*	0.861	0.860	0.501	0.500	0.500	0.501
	test	0.849	0.859	0.500	0.501	0.500	0.499
MNIST	*train*	1.000	0.998	0.107	0.129	0.243	0.330
	test	0.973	0.987	0.106	0.132	0.251	0.331
LFW	*train*	1.000	0.999	0.109	0.137	0.214	0.428
	test	0.842	0.835	0.116	0.119	0.200	0.463
CIFAR10	*train*	0.999	0.996	0.100	0.098	0.103	0.100
	test	0.621	0.664	0.101	0.100	0.105	0.093
CIFAR100	*train*	0.999	0.977	0.010	0.010	0.010	0.011
	test	0.257	0.312	0.010	0.010	0.011	0.011

Table C.2: Train and test accuracies for models trained with either dropout or (ϵ, δ)-differential privacy for various values of ϵ.

or the accuracy of the private model was significantly lower than that of the unprotected model.

Abadi et al. [1] mitigate the high cost in accuracy by first pre-training on public data, and then fine-tuning only the top layers with differential privacy on the private training set. While this public transfer learning approach may not always be possible, it has two key benefits, the first being that the resulting model's performance is far less poor. Second, only the final layers of such a model are trained on the private data, and thus our attack may only be able to effectively target those layers. Our experiments [83] show that our attack is far more effective when all layers are leveraged, and that the earlier layers often account for a sizable portion of the information leakage. This suggests that, when possible, a transfer learning scheme like that of Abadi et al. could be a practical defense.

Regularization

Given the connection between membership inference and overfitting, regularization, such as dropout [138], which aims to reduce overfitting, has also been proposed to combat membership inference. Generalization alone is not sufficient to protect against membership inference [161], and in fact, our empirical results (Chapter 8, Section 8.3.3) show that we can successfully attack even models with negligible generalization error; however, dropout has been shown not only to reduce overfitting, but to strengthen privacy guarantees in neural networks [66]. Figure C.1 shows the accuracy of our attack with and without dropout. We find that dropout does not significantly impact the accuracy of our attack in most cases. However, as opposed to DP, dropout is typically *beneficial* to the performance of the model, while providing a modest defense. In this light, regularization (including dropout) may in fact be the more practical defensive measure, insofar as it improves test accuracy, because

better generalization does appear to make membership more difficult, though clearly not impossible, for an attacker.

Still, we warn that this may not be universally true of all forms of regularization, even regularization that improves generalization—as we have demonstrated, a model can still leak membership information through its parameters while making correct predictions on unseen points.

Defenses in the Black-box Setting

For membership inference in the black-box setting, Shokri et al. [128] also propose a number of other possible defenses, such as restricting the prediction vector to the top k classes, or increasing the entropy of the prediction vector via increasing the normalization temperature of the softmax. However, these defenses are easily circumvented in the white-box setting, as the pre-modified outputs are still available to an attacker in this threat model.

Similarly, Salem et al. [121] propose a defense called *model stacking*, in which two models are trained separately on the training data and a third model makes predictions based on the outputs of the first two. While Salem et al. found this to be an effective defense against black-box approaches, this defense is likewise circumvented in the white-box setting, as the initial two models are available to the attacker.

Appendix D

Further Details on Reconstruction-based Property Inference

D.1 Additional Imagenet Reconstructions

Figure D.1 shows additional ImageNet feature visualizations using more sophisticated techniques than Figure 8.10. In particular, we perform the optimization using a 2D FFT parameterization of the image, rather than optimization directly in pixel space—this technique was suggested by Olah et al. [109]. This FFT preconditioning approach has the benefit of de-correlating pixels, allowing for larger step sizes and faster convergence. We also use random rotations, which Olah et al. found improved perceptual quality of feature visualizations for individual neurons. This encourages reconstructions that are structurally invariant to rotation.

We also add i.i.d. Gaussian noise to the model parameters each step of the optimization process (resetting the parameters each time to maintain stationarity), which was inspired by the observation of Terzi et al. [143] that adding noise scaled by the inverse Fisher Information Matrix of the model parameters once at the beginning of training improves reconstruction quality. This iterative noise emphasizes the most significant model parameters with respect to the classification loss that we optimize, which is conceptually similar to global weight

Figure D.1: Similar reconstructions to Figure 8.10, except that more sophisticated feature visualization techniques are used.

pruning with a fixed threshold. Intuitively, the idea is to drown out the contribution of neurons that are only weakly associated (in terms of input gradients) with a particular classification output.

D.2 Set Transformer Details

The set transformer we use is a direct TensorFlow adaptation of the PyTorch implementation[1] provided by Lee et al. [81]. In particular, the encoder is a stack of two Induced Set Attention Blocks (ISABs) [81, Eq. (14)] and the decoder is a row-wise Feed-Forward network (rFF) followed by a Set Attention Block (SAB) and finally a Pooling Multi-headed Attention (PMA) layer [81, Eq. (15)-(16)]. We use the same default parameters as the authors' implementation,[2] which uses a hidden dimension of 128 and 4 attention heads. The PMA block uses 8 seed vectors. The only difference is that our decoder reduces the output to a single vector of logits rather than a set of multiple vectors, as would be done in set-to-set translation.

[1] `https://github.com/juho-lee/set_transformer`
[2] `https://github.com/juho-lee/set_transformer/blob/master/models.py`

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. *Deep Learning with Differential Privacy.* In ACM Conference on Computer and Communications Security (CCS), 2016.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. *Sanity Checks for Saliency Maps.* In Neural Information Processing Systems (NIPS), 2018.

[3] Guillaume Alain and Yoshua Bengio. *Understanding Intermediate Layers Using Linear Classifier Probes.* In Neural Information Processing Systems (NIPS), 2016.

[4] Cem Anil, James Lucas, and Roger Gross. *Sorting Out Lipschitz Function Approximation.* In International Conference on Machine Learning (ICML), 2019.

[5] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. *Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers.* 2015.

[6] Robert J. Aumann and Lloyd S. Shapley. *Values of Non-Atomic Games.* Princeton University Press, 1974.

[7] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. *On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation.* PLOS One, 2015.

[8] Mislav Balunovic and Martin Vechev. *Adversarial Training and Provable Defenses: Bridging the Gap.* In International Conference on Learning Representations (ICLR), 2020.

[9] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. *Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives.* In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.

[10] Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. *Smooth Loss Functions for Deep Top-k Classification.* In International Conference on Learning Representations (ICLR), 2018.

[11] Åke Björck and C. Bowie. *An Iterative Algorithm for Computing the Best Estimate of an Orthogonal Matrix.* In SIAM Journal on Numerical Analysis, 1971.

[12] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. *Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings*, 2016.

[13] Justin Brickell and Vitaly Shmatikov. *The Cost of Privacy: Destruction of Data-mining Utility in Anonymized Data Publishing*. In International Conference on Knowledge Discovery and Data Mining (KDD), 2008.

[14] Andrew Brock, Soham De, and Samuel L. Smith. *Characterizing Signal Propagation to Close the Performance Gap in Unnormalized ResNets*. In International Conference on Learning Representations (ICLR), 2021.

[15] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. *High-Performance Large-Scale Image Recognition Without Normalization*. In International Conference on Machine Learning (ICML), 2021.

[16] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. *Adversarial Patch*. In NIPS Workshop on Machine Learning and Computer Security, 2017.

[17] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. *A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks*, 2017.

[18] Kaylee Burns, Lisa Anne Hendricks, Trevor Darrell, and Anna Rohrbach. *Women also Snowboard: Overcoming Bias in Captioning Models*, 2018.

[19] Nicholas Carlini and David Wagner. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*. In ACM Workshop on Artificial Intelligence and Security, 2017.

[20] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. *Property Inference From Poisoning*, 2021.

[21] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. *Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks*. In IEEE Winter Conference on Applications of Computer Vision, 2018.

[22] Kamalika Chaudhuri and Claire Monteleoni. *Privacy-preserving Logistic Regression*. In Neural Information Processing Systems (NIPS), 2009.

[23] Nitesh Chawla. *Data Mining for Imbalanced Datasets: An Overview*. In Oded Maimon and Lior Rokach, editors, Data Mining and Knowledge Discovery Handbook, chapter 40. Springer, 2010.

[24] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. *Parseval Networks: Improving Robustness to Adversarial Examples*. In International Conference on Machine Learning (ICML), 2017.

[25] Jeremy Cohen, Todd Huster, and Ra Cohen. *Universal Lipschitz Approximation in Bounded Depth Neural Networks*, 2019.

[26] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. *Certified Adversarial Robustness via Randomized Smoothing*. In International Conference on Machine Learning (ICML), 2019.

[27] Graham Cormode. *Personal Privacy vs Population Privacy: Learning to Attack Anonymization*. In International Conference on Knowledge Discovery and Data Mining (KDD), 2011.

[28] Francesco Croce and Matthias Hein. *Provable Robustness Against All Adversarial l_p-perturbations for $p \geqslant 1$*. In International Conference on Learning Representations (ICLR), 2020.

[29] Francesco Croce and Matthias Hein. *Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks*. In International Conference on Machine Learning (ICML), 2020.

[30] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. *Provable Robustness of ReLU networks via Maximization of Linear Regions.* In International Conference on Artificial Intelligence and Statistics (AISTATS), 2019.

[31] Anupam Datta, Shayak Sen, and Yair Zick. *Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems.* In IEEE Symposium on Security and Privacy (S&P), 2016.

[32] Anupam Datta, Matt Fredrikson, Klas Leino, Kaiji Lu, Zifan Wang, Ricardo Shih, and Shayak Sen. *Exploring Conceptual Soundness with TruLens.* In Proceedings of Machine Learning Research, 2022.

[33] Cynthia Dwork. *Differential privacy.* In International Colloquium on Automata, Languages and Programming (ICALP), 2006.

[34] Cynthia Dwork, Aaron Roth, et al. *The Algorithmic Foundations of Differential Privacy.* Found. Trends Theor. Comput. Sci., 9(3-4):211–407, 2014.

[35] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. *Exposed! A Survey of Attacks on Private Data.* Annual Review of Statistics and Its Application, 2017.

[36] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. *A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations.* In NIPS Workshop on Machine Learning and Computer Security, 2017.

[37] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola Schoenlieb. *On the Connection Between Adversarial Robustness and Saliency Map Interpretability.* In International Conference on Machine Learning (ICML), 2019.

[38] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. *Robust Physical-World Attacks on Machine Learning Models.* In Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[39] Farzan Farnia, Jesse Zhang, and David Tse. *Generalizable Adversarial Training via Spectral Normalization.* In International Conference on Learning Representations (ICLR), 2019.

[40] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. *Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks.* In Neural Information Processing Systems (NIPS), 2019.

[41] Ruth Fong and Andrea Vedaldi. *Interpretable Explanations of Black Boxes by Meaningful Perturbation.* In International Conference on Computer Vision (ICCV), 2017.

[42] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. *Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing.* In USENIX Security Symposium, 2014.

[43] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. *Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures.* In ACM Conference on Computer and Communications Security (CCS), 2015.

[44] Aymeric Fromherz, Klas Leino, Matt Fredrikson, Bryan Parno, and Corina Păsăreanu. *Fast Geometric Projections for Local Robustness Certification.* In International Conference on Learning Representations (ICLR), 2021.

[45] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. *Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations.* In ACM Conference on Computer and Communications Security (CCS), 2018.

[46] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. *ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.* In International Conference on Learning Representations (ICLR), 2019.

[47] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L. Beam. *The False Hope of Current Approaches to Explainable Artificial Intelligence in Health Care.* In The Lancet Digital Health, 2021.

[48] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*, 2015.

[49] Divya Gopinath, Guy Katz, Corina S. Păsăreanu, and Clark Barrett. *DeepSafe: A Data-Driven Approach for Assessing Robustness of Neural Networks.* In Automated Technology for Verification and Analysis, 2018.

[50] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. *Regularisation of Neural Networks by Enforcing Lipschitz Continuity.* Machine Learning, 110(2):393–416, 2021.

[51] Sven Gowal, Krishnamurthy (Dj) Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. *Scalable Verified Training for Provably Robust Image Classification.* In International Conference on Computer Vision (ICCV), 2019.

[52] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. *Identifying Personal Genomes by Surname Inference.* In Science, 2013.

[53] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron C. Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. *Brain Tumor Segmentation with Deep Neural Networks*, 2015.

[54] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. *LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks*, 2017.

[55] Haibo He and Edwardo A. Garcia. *Learning from Imbalanced Data.* In IEEE Transactions on Knowledge and Data Engineering, 2009.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* In International Conference on Computer Vision (ICCV), 2015.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Identity mappings in deep residual networks.* In European Conference on Computer Vision (ECCV), 2016.

[58] Matthias Hein and Maksym Andriushchenko. *Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation.* In Neural Information Processing Systems (NIPS), 2017.

[59] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. *Introducing EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification.* In IEEE International Geoscience and Remote Sensing Symposium, 2018.

[60] Jonathan Helland and Nathan VanHoudnos. *On the Human-recognizability Phenomenon of Adversarially Trained Deep Image Classifiers*, 2020.

[61] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. *Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning*, 2017.

[62] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. *Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays.* In PLoS Genetics, 2008.

[63] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical report, University of Massachusetts, Amherst, 2007.

[64] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. *Limitations of the Lipschitz Constant as a Defense Against Adversarial Examples.* In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2018.

[65] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. *Adversarial Examples Are Not Bugs, They Are Features.* In Neural Information Processing Systems (NIPS), 2019.

[66] Prateek Jain, Vivek Kulkarni, Abhradeep Thakurta, and Oliver Williams. *To Drop or Not to Drop: Robustness, Consistency and Differential Privacy Properties of Dropout*, 2015.

[67] Bargav Jayaraman and David Evans. *Evaluating Differentially Private Machine Learning in Practice.* In USENIX Security Symposium, 2019.

[68] Jinyuan Jia, Xiaoyu Cao, Binghui Wang, and Neil Zhenqiang Gong. *Certified Robustness for Top-k Predictions against Adversarial Perturbations via Randomized Smoothing.* In International Conference on Learning Representations (ICLR), 2020.

[69] Matt Jordan, Justin Lewis, and Alexandros G. Dimakis. *Provable Certificates for Adversarial Examples: Fitting a Ball in the Union of Polytopes.* In Neural Information Processing Systems (NIPS), 2019.

[70] Kyle D. Julian, Jessica Lopez, Jeffrey S. Brush, Michael P. Owen, and Mykel J. Kochenderfer. *Policy Compression for Aircraft Collision Avoidance Systems.* In IEEE/AIAA Digital Avionics Systems Conference (DASC), 2016.

[71] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks.* In International Conference on Computer-Aided Verification (CAV), 2017.

[72] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. *The Marabou Framework for Verification and Analysis of Deep Neural Networks.* In International Conference on Computer Aided Verification (CAV), 2019.

[73] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. *Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)*, 2018.

[74] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. *Investigating the Influence of Noise and Distractors on the Interpretation of Neural Networks.* In Neural Information Processing Systems (NIPS), 2016.

[75] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* In International Conference on Learning Representations (ICLR), 2015.

[76] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images.* Technical report, 2009.

[77] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. *Adversarial Examples in the Physical World*, 2016.

[78] Y. Le and X. Yang. *Tiny ImageNet Visual Recognition Challenge*. Technical report, 2015.

[79] Yann LeCun, Corinna Cortes, and CJ Burges. *MNIST Handwritten Digit Database*. 2010. URL `http://yann.lecun.com/exdb/mnist`.

[80] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. *Certified Robustness to Adversarial Examples with Differential Privacy*. In IEEE Symposium on Security and Privacy (S&P), 2018.

[81] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. *Set Transformer: A Framework for Attention-based Permutation-invariant Neural Networks*. In International Conference on Machine Learning (ICML), 2019.

[82] Sungyoon Lee, Jaewook Lee, and Saerom Park. *Lipschitz-Certifiable Training with a Tight Outer Bound*. In Neural Information Processing Systems (NIPS), 2020.

[83] Klas Leino and Matt Fredrikson. *Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference*. In USENIX Security Symposium, 2020.

[84] Klas Leino and Matt Fredrikson. *Relaxing Local Robustness*. In Neural Information Processing Systems (NIPS), 2021.

[85] Klas Leino, Shayak Sen, Anupam Datta, and Matt Fredrikson. *Influence-Directed Explanations for Deep Convolutional Networks*. In IEEE International Test Conference (ITC), 2018.

[86] Klas Leino, Emily Black, Matt Fredrikson, Shayak Sen, and Anupam Datta. *Feature-Wise Bias Amplification*. In International Conference on Learning Representations (ICLR), 2019.

[87] Klas Leino, Aymeric Fromherz, Ravi Mangal, Matt Fredrikson, Bryan Parno, and Corina Păsăreanu. *Self-Repairing Neural Networks: Provable Safety for Deep Networks via Dynamic Repair*, 2021.

[88] Klas Leino, Zifan Wang, and Matt Fredrikson. *Globally-Robust Neural Networks*. In International Conference on Machine Learning (ICML), 2021.

[89] Klas Leino, Jonathan Helland, Saranya Vijayakumar, Matt Fredrikson, and Nathan VanHoudnos. *Robust Features Can Leak Instances and Their Properties*, 2022.

[90] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. *Membership Privacy: A Unifying Framework For Privacy Definitions*. In ACM Conference on Computer and Communications Security (CCS), 2013.

[91] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Joern-Henrik Jacobsen. *Preventing Gradient Attenuation in Lipschitz Constrained Convolutional Networks*. In Neural Information Processing Systems (NIPS), 2019.

[92] Dong C. Liu and Jorge Nocedal. *On the Limited Memory BFGS Method for Large Scale Optimization*. In Mathematical Programming, 1989.

[93] Yunhui Long, Vincent Bindschaedler, and Carl A. Gunter. *Towards Measuring Membership Privacy*, 2017.

[94] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. *Understanding Membership Inferences on Well-Generalized Learning Models*, 2018.

[95] Diego Manzanas Lopez, Taylor Johnson, Hoang-Dung Tran, Stanley Bak, Xin Chen, and Kerianne L. Hobbs. *Verification of Neural Network Compression of ACAS Xu Lookup Tables with Star Set Reachability.* In AIAA Scitech Forum, 2021.

[96] Kaiji Lu, Piotr Mardziel, Klas Leino, Matt Fedrikson, and Anupam Datta. *Influence Paths for Characterizing Subject-Verb Number Agreement in LSTM Language Models.* In Association for Computational Linguistics (ACL), 2020.

[97] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks.* In International Conference on Learning Representations (ICLR), 2018.

[98] Marcus A. Maloof. *Learning When Data Sets are Imbalanced and When Costs are Unequeal and Unknown.* In ICML Workshop on Learning from Imbalanced Data Sets, 2003.

[99] Maciej A. Mazurowski, Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker, and Georgia D. Tourassi. *Training Neural Network Classifiers for Medical Decision Making: The Effects of Imbalanced Datasets on Classification Performance.* Neural Networks, 2008.

[100] H. Brendan McMahan and Galen Andrew. *A General Approach to Adding Differential Privacy to Iterative Training Procedures*, 2018.

[101] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. *Inference Attacks Against Collaborative Learning*, 2018.

[102] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. *Exploiting unintended feature leakage in collaborative learning.* In IEEE Symposium on Security and Privacy (S&P), 2019.

[103] Matthew Mirman, Timon Gehr, and Martin Vechev. *Differentiable Abstract Interpretation for Provably Robust Neural Networks.* In International Conference on Machine Learning (ICML), 2018.

[104] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. *Spectral Normalization for Generative Adversarial Networks.* In International Conference on Learning Representations (ICLR), 2018.

[105] Kevin P. Murphy. *Gaussian Classifiers.* University Lecture, 2007. URL `https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/gaussClassif.pdf`.

[106] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012. ISBN 0262018020, 9780262018029.

[107] Milad Nasr, Reza Shokri, and Amir Houmansadr. *Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks*, 2018.

[108] 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.

[109] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. *Feature Visualization.* 2017.

[110] Thomas Oommen, Laurie Baise, and Richard Vogel. *Sampling Bias and Class Imbalance in Maximum-likelihood Logistic Regression.* In Mathematical Geosciences, 2011.

[111] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. *The Limitations of Deep Learning in Adversarial Settings.* In European Symposium on Security and Privacy, 2016.

[112] Mathias PM Parisot, Balazs Pejo, and Dayana Spagnuelo. *Property Inference Attacks on Convolutional Neural Networks: Influence and Implications of Target Model's Complexity*, 2021.

[113] Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. *Resurrecting the Sigmoid in Deep Learning through Dynamical Isometry: Theory and Practice.* In Neural Information Processing Systems (NIPS), 2017.

[114] Vitali Petsiuk, Abir Das, and Kate Saenko. *RISE: Randomized Input Sampling for Explanation of Black-box Models.* In British Machine Vision Conference, 2019.

[115] John Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization.* In Advances in Kernel Methods: Support Vector Learning, 1995.

[116] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. *Knock Knock, Who's There? Membership Inference on Aggregate Location Data*, 2017.

[117] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. *SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability.* In Neural Information Processing Systems (NIPS), 2017.

[118] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier.* In International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[119] David Rolnick and Konrad Kording. *Reverse-engineering deep ReLU networks.* In International Conference on Machine Learning (ICML), 2020.

[120] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. *Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning*, 2009.

[121] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. *ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models.* In Annual Network and Distributed System Security Symposium (NDSS), 2019.

[122] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Bach, and Klaus-Robert Müller. *Evaluating the Visualization of What a Deep Neural Network has Learned.* In IEEE Transactions on Neural Networks and Learning Systems, 2017.

[123] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. *Genomic Privacy and Limits of Individual Detection in a Pool.* In Nature Genetics, 2009.

[124] Hanie Sedghi, Vineet Gupta, and Philip M. Long. *The Singular Values of Convolutional Layers.* In International Conference on Learning Representations (ICLR), 2019.

[125] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. *Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization.* In International Conference on Computer Vision (ICCV), 2017.

[126] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. *Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition.* In ACM Conference on Computer and Communications Security (CCS), 2016.

[127] Reza Shokri and Vitaly Shmatikov. *Privacy-Preserving Deep Learning.* In ACM Conference on Computer and Communications Security (CCS), 2015.

[128] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. *Membership Inference Attacks against Machine Learning Models.* In IEEE Symposium on Security and Privacy (S&P), 2016.

[129] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. *Learning Important Features Through Propagating Activation Differences.* 2017.

[130] Suyash S. Shringarpure and Carlos D. Bustamante. *Privacy Risks from Genomic Data-Sharing Beacons.* In The American Journal of Human Genetics, 2015.

[131] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, 2014.

[132] Sahil Singla and Soheil Feizi. *Bounding Singular Values of Convolution Layers*, 2019.

[133] Aman Sinha, Hongseok Namkoong, and John Duchi. *Certifiable Distributional Robustness with Principled Adversarial Training.* In International Conference on Learning Representations (ICLR), 2018.

[134] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. *SmoothGrad: Removing Noise by Adding Noise*, 2017.

[135] Congzheng Song and Vitaly Shmatikov. *Overlearning Reveals Sensitive Attributes*, 2019.

[136] Liwei Song, Reza Shokri, and Prateek Mittal. *Privacy Risks of Securing Machine Learning Models Against Adversarial Examples.* In ACM Conference on Computer and Communications Security (CCS), 2019.

[137] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. *Striving for Simplicity: The All Convolutional Net*, 2015.

[138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* In Journal of Machine Learning Research, 2014.

[139] Pierre Stock and Moustapha Cissé. *ConvNets and ImageNet Beyond Accuracy: Explanations, Bias Detection, Adversarial Examples and Model Criticism*, 2017.

[140] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks.* In International Conference on Machine Learning (ICML), 2017.

[141] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. *Intriguing Properties of Neural Networks.* In International Conference on Learning Representations (ICLR), 2014.

[142] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. *Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12.* 2017.

[143] Matteo Terzi, Alessandro Achille, Marco Maggipinto, and Gian Antonio Susto. *Adversarial Training Reduces Information and Improves Transferability*, 2020.

[144] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. *Evaluating Robustness of Neural Networks with Mixed Integer Programming.* In International Conference on Learning Representations (ICLR), 2019.

[145] Florian Tramèr and Dan Boneh. *Adversarial Training and Robustness for Multiple Perturbations.* In Neural Information Processing Systems (NIPS), 2019.

[146] Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Jörn-Henrik Jacobsen. *Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations.* In International Conference on Machine Learning (ICML), 2020.

[147] Asher Trockman and J Zico Kolter. *Orthogonalizing Convolutional Layers with the Cayley Transform.* In International Conference on Learning Representations (ICLR), 2021.

[148] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. *Robustness May Be at Odds with Accuracy.* In International Conference on Learning Representations (ICLR), 2019.

[149] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. *Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks.* In Neural Information Processing Systems (NIPS), 2018.

[150] Byron C. Wallace, Kevin Small, Carla E. Brodley, and Thomas A. Trikalinos. *Class Imbalance, Redux.* In International Conference on Data Mining, 2011.

[151] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. *Score-CAM: Improved Visual Explanations Via Score-Weighted Class Activation Mapping.* In IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020.

[152] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. *Learning Your Identity and Disease from Research Papers: Information Leaks in Genome Wide Association Studies.* In ACM Conference on Computer and Communications Security (CCS), 2009.

[153] Zifan Wang, Haofan Wang, Shakul Ramkumar, Matt Fredrikson, Piotr Mardziel, and Anupam Datta. *Smoothed Geometry for Robust Attribution.* In Neural Information Processing Systems (NIPS), 2020.

[154] Zifan Wang, Matt Fredrikson, and Anupam Datta. *Boundary Attributions Provide Normal (Vector) Explanations*, 2021.

[155] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. *Towards Fast Computation of Certified Robustness for ReLU Networks.* In International Conference on Machine Learning (ICML), 2018.

[156] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. *Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach.* In International Conference on Learning Representations (ICLR), 2018.

[157] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. *Scaling Provable Adversarial Defenses.* In Neural Information Processing Systems (NIPS), 2018.

[158] Xi Wu, Matt Fredrikson, Wentao Wu, Somesh Jha, and Jeffrey F. Naughton. *Revisiting Differentially Private Regression: Lessons From Learning Theory and their Consequences*, 2015.

[159] Kai Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. *Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability.* In International Conference on Learning Representations (ICLR), 2019.

[160] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R. Salakhutdinov, and Kamalika Chaudhuri. *A Closer Look at Accuracy vs. Robustness.* In Neural Information Processing Systems (NIPS), 2020.

[161] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. *Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting.* In IEEE Computer Security Foundations Symposium (CSF), 2018.

[162] Samuel Yeom, Irene Giacomelli, Alan Menaged, Matt Fredrikson, and Somesh Jha. *Overfitting, Robustness, and Malicious Algorithms: A Study of Potential Causes of Privacy Risk in Machine Learning*, 2020.

[163] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. *How Transferable Are Features in Deep Neural Networks?* In Neural Information Proceessing Systems (NIPS), 2014.

[164] H. Peyton Young. *Individual Contribution and Just Compensation.* In Alvin E. Roth, editor, The Shapley Value, chapter 17, pages 267–278. Cambridge University Press, 1988.

[165] Matthew D. Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks.* In European Conference on Computer Vision (ECCV), 2014.

[166] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. *Theoretically Principled Trade-off between Robustness and Accuracy.* In International Conference on Machine Learning (ICML), 2019.

[167] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. *Fixup Initialization: Residual Learning Without Normalization.* In International Conference on Learning Representations (ICLR), 2019.

[168] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. *Efficient Neural Network Robustness Certification with General Activation Functions.* In Neural Information Processing Systems (NIPS). 2018.

[169] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. *Towards Stable and Efficient Training of Verifiably Robust Neural Networks.* In International Conference on Learning Representations (ICLR), 2020.

[170] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. *Top-down Neural Attention by Excitation Backprop.* In European Conference on Computer Vision (ECCV), 2016.

[171] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. *Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints*, 2017.

[172] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. *Learning Deep Features for Discriminative Localization.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[173] Dongmian Zou, Radu Balan, and Maneesh Singh. *On Lipschitz Bounds of General Convolutional Neural Networks.* 2019.