UNIVERSITY OF CALIFORNIA SAN DIEGO

A Principled Approach to Trustworthy Machine Learning

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Yao-Yuan Yang

Committee in charge:

Professor Kamalika Chaudhuri, Chair
Professor Sanjoy Dasgupta
Professor Sicun Gao
Professor Tara Javidi
Professor Lawrence Saul

2022

The dissertation of Yao-Yuan Yang is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

EPIGRAPH

falsely believing that the answers to all scientific questions reside in the data, to be unveiled through clever data-mining tricks. Much of this data-centric history still haunts us today. We live in an era that presumes Big Data to be the solution to all our problems. Courses in "data science" are proliferating in our universities, and jobs for "data scientists" are lucrative in the companies that participate in the "data economy." But I hope with this book to convince you that data are profoundly dumb.

—— Judea Pearl, The Book of Why: The New Science of Cause and Effect

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Kamalika Chaudhuri, who guided me to become an (almost) independent researcher. She teaches me about all aspects of research, from finding research ideas, coming up with hypotheses, designing/executing experiments to clearly presenting research findings, and promoting my work. I really appreciate her patient guidance in my writing communication and her constructive career advice. During my five years at UCSD, she has been tremendously supportive and enlightening. I feel incredibly fortunate to have her as my Ph.D. advisor.

I also want to thank the rest of my committee, Sanjoy Dasgupta, Sicun Gao, Tara Javidi, and Lawrence Saul, for their helpful feedback and comments that helped me complete my thesis and dissertation.

I convey my sincere gratitude to Cyrus Rashtchian, who is a collaborator in many of my research projects. I really enjoyed the thoughtful research discussion we had, and I really learned a lot about research during our collaboration, including good communication and experiment design.

I want to express my immense appreciation to Angel Hsing-Chi Hwang for being a supportive, loving, considerate, and thoughtful partner for the past several years. I must thank you for your full support when I am going through rough times. In addition, I enjoy all of our time spent together traveling, exploring new restaurants, and working on multiple research projects! This really helps me fuel up and get ready for new challenges. Your companionship has also helped me grow as a person, brought color to my life, and made my Ph.D. journey more delightful.

My sincere thanks also go to Chi-Ning Chou for friendship and fun discussions across a wide spectrum of research topics (from neuroscience, evolutionary dynamics to quantum computing). Although the content of our recent collaboration did not make it to this thesis, I am still glad to be able to work with you. In addition, I really love the neuroscience reading group you and Brabeeba Wang held during the Covid-19 pandemic. I really gain a lot of knowledge

from the reading group.

I am grateful to all my additional collaborators – a partial list includes Michal Moshkovitz, Ruslan Salakhutdinov, Yizhen Wang, and Hongyang Zhang. Without you, many of my research projects would not have been successful. Although our paper did not make it to this thesis, I want to thank Bogdan Kulynych, Yaodong Yu, Jarosław Błasiok, and Preetum Nakkiran for the wonderful journey of exploring the relationship between differential privacy and distributional generalization.

I would like to thank my office mates – Julaiti Alafate, Robi Bhattacharjee, Joseph Geumlek, Jacob Imola, Tatsuki Koga, Zhifeng Kong, Casey Meehan, Mary Ann Smart, Yizheng Wang, Zhi Wang, Chhavi Yadav, Songbai Yan for the great time we spent in our office.

I want to thank all my friends that are around me during the years of pursuing my Ph.D. I thank Chih-Kuan Yeh for many interesting discussions on different research ideas and job hunting advice. I also thank Yu-Kai Su, Jiunn-Chia Huang, and Kuan-Hao Huang for hosting me when I travel. Additionally, I want to thank Chih-Hui Ho, Yi-An Lai, Ping-Min Lin, Kurtis Liu, Shu-Ting Wang, Victor Wang, and Chun-Han Yao for their friendship.

Going back to my time at National Taiwan University, I want to acknowledge Hsuan-Tien Lin, who is my undergraduate research advisor, for leading me into the machine learning research path. He teaches me the basics of doing research, including carrying out a research project, academic writing, and presenting.

Last but not least, I want to thank my parents, Jui-Chen Yu and Hung-Jen Yang, for their unconditional support in my life.

Chapter 3 is based on the manuscript "A Closer Look at Accuracy vs. Robustness" by Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri [240]. The manuscript is published in the *Conference on Neural Information Processing Systems*, 2020. The dissertation author is the co-primary researcher and co-author of the paper.

Chapter 4 is based on the manuscript "Robustness for Non-Parametric Classification: A Generic Attack and Defense" by Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri [239]. The manuscript is published in the *International Conference on Artificial Intelligence and Statistics*, 2020. The dissertation author is the co-primary researcher and co-author of the paper.

Chapter 5 is based on the manuscript "Connecting Interpretability and Robustness in Decision Trees through Separation" by Michal Moshkovitz, Yao-Yuan Yang, and Kamalika Chaudhuri [152]. The manuscript is published in the *International Conference on Machine Learning*, 2021. The dissertation author is the co-author of the paper.

Chapter 6 is based on the manuscript "Robustness and Generalization to Nearest Categories" by Yao-Yuan Yang, Cyrus Rashtchian, Ruslan Salakhutdinov, and Kamalika Chaudhuri [238]. The manuscript is currently in submission. The dissertation author is the primary researcher and co-author of the paper.

Chapter 7 is based on the manuscript "Understanding Rare Spurious Correlations in Neural Networks" by Yao-Yuan Yang and Kamalika Chaudhuri [235]. The manuscript is currently in submission. The dissertation author is the primary researcher and co-author of the paper.

VITA

2012-2016	B. S. in Computer Science, National Taiwan University, Taiwan
2017-2022	Ph. D. in Computer Science, University of California San Diego

PUBLICATIONS

B. Kulynych*, **Y.-Y. Yang***, Y. Yu, J. Błasiok, and P. Nakkiran What You See is What You Get: Distributional Generalization for Algorithm Design in Deep Learning, arXiv preprint arXiv:2204.03230, 2022

Y.-Y. Yang, and K. Chaudhuri. Understanding Rare Spurious Correlations in Neural Networks. arXiv preprint arXiv:2202.05189, 2022.

Y.-Y. Yang*, M. Hira*, Z. Ni*, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi. TorchAudio: Building Blocks for Audio and Speech Processing, ICASSP, 2022

A. H.-C. Hwang, C. Y. Wang, **Y.-Y. Yang**, and A. S. Won. Hide and seek: Choices of Virtual Backgrounds in Video Chats and Their Effects on Perception, CSCW, 2021

Y.-Y. Yang, C. Rashtchian, R. Salakhutdinov, and K. Chaudhuri. Probing Predictions on OOD Images via Nearest Categories, arXiv preprint arXiv:2011.08485, 2020

M. Moshkovitz, **Y.-Y. Yang**, and K. Chaudhuri. Connecting Interpretability and Robustness in Decision Trees through Separation, ICML, 2021

Y.-Y. Yang*, C. Rashtchian*, H. Zhang, R. Salakhutdinov, and K. Chaudhuri. A Closer Look at Accuracy vs. Robustness, NeurIPS, 2020

Y.-Y. Yang*, C. Rashtchian*, Y. Wang, and K. Chaudhuri. Robustness for Non-Parametric Classification: A Generic Attack and Defense, AISTATS, 2020

B. Cosman, M. Endres, G. Sakkas, L. Medvinsky, **Y.-Y. Yang**, R. Jhala, K. Chaudhuri, and W. Weimer. PABLO: Helping Novices Debug Python Code Through Data-Driven Fault Localization, SIGCSE, 2020

Y.-Y. Yang, Y.-A. Lin, H.-M. Chu, and H.-T. Lin. Deep Learning with a Rethinking Structure for Multi-label Classification, ACML, 2019.

Y.-Y. Yang, K.-H. Huang, C.-W. Chang, and H.-T. Lin. Cost-Sensitive Reference Pair Encoding for Multi-Label Learning, PAKDD, 2018

* equal contribution.

ABSTRACT OF THE DISSERTATION

A Principled Approach to Trustworthy Machine Learning

by

Yao-Yuan Yang

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Kamalika Chaudhuri, Chair

Traditional machine learning operates under the assumption that training and testing data are drawn independently from the same distribution. However, this assumption does not always hold. In this thesis, we take a principled approach toward three major challenges in settings where this assumption fails to hold – i) robustness to adversarial inputs, ii) handling unseen examples during test time, and iii) avoiding learning spurious correlations.

We study what happens when small adversarial perturbations are made to the inputs. We investigate neural networks, which frequently operate on natural datasets such as images. We find that in these datasets, differently labeled examples are often far away from each other. Under this condition, we prove that a perfectly robust and accurate classifier exists, suggesting that

there is no intrinsic tradeoff between adversarial robustness and accuracy on these datasets. Next, we look into non-parametric classifiers, which operate on datasets without such a separation condition. We design a defense algorithm – adversarial pruning – that successfully improves the robustness of many non-parametric classifiers, including k-nearest neighbor, decision tree, and random forest. Adversarial pruning can also be seen as a finite sample approximation to the classifier with the highest accuracy under robustness constraints. Finally, we connect robustness and interpretability on decision trees by designing an algorithm that is guaranteed to achieve good accuracy, robustness, and interpretability when the data is linearly separated.

We study what happens when small adversarial perturbations are made to the inputs. We investigate neural networks, which frequently operate on natural datasets such as images. We find that in these datasets, differently labeled examples are often far away from each other. Under this condition, we prove that a perfectly robust and accurate classifier exists, suggesting that there is no intrinsic tradeoff between adversarial robustness and accuracy on these datasets. Next, we look into non-parametric classifiers, which operate on datasets without such a separation condition. We design a defense algorithm – adversarial pruning – that successfully improves the robustness of many non-parametric classifiers, including k-nearest neighbor, decision tree, and random forest. Adversarial pruning can also be seen as a finite sample approximation to the classifier with the highest accuracy under robustness constraints. Finally, we connect robustness and interpretability on decision trees by designing an algorithm that is guaranteed to achieve good accuracy, robustness, and interpretability when the data is linearly separated.

Next, we explore what would happen if examples that do not belong in the training set, i.e., out-of-distribution (OOD) examples, are given to a model as input during testing time. We identify that neural networks tend to predict OOD inputs as the label of the closest training example, and adversarially robust networks amplify this behavior. These findings can shed light on many long-standing questions surrounding generalization, including how adversarial robust training methods change the decision boundary, why an adversarially robust network performs

better on corrupted data, and when OOD examples can be hard to detect.

Finally, we investigate the case where a few examples of a certain class with spurious features are presented during training. We find that merely three of these spurious examples can cause the network to learn a spurious correlation. Our result suggests that neural networks are highly sensitive to small amounts of training data. Although this feature enables efficient learning, it also results in rapid learning of spurious correlation.

Chapter 1

Introduction

The *statistical learning theory framework* [27] has been the cornerstone of many traditional machine learning models. The core assumption of this framework is that data in the training and testing sets are sampled independently from exactly the same distribution (i.i.d.). Operating under this framework yields fruitful theoretical results, and many algorithms are guaranteed to perform well. For instance, the k-nearest neighbor classifier (with appropriate k) is proven to converge to the optimal classifier when the training set is large enough. However, as machine learning models are being adopted in more and more scenarios, the i.i.d. assumption may no longer be sufficient.

Trustworthy ML studies how machine learning models behave under various settings where the statistical learning theory framework assumption does not hold. In this thesis, we take a principled approach toward three of these settings, which includes i) adversarial robustness [83, 144], where there exists an adversary that can modify test time inputs, ii) out-of-distribution generalization [56, 68, 95], where examples from unseen classes may appear during test time, and iii) spurious correlation [112, 189, 207, 213], where during test time, distribution may shift, and the spurious feature that a classifier depends on may disappear. It is critical to understand these behaviors to ensure the reliability of machine learning models at deployment time, which

motivates the subject of the present thesis.

1.1 Adversarial Robustness

Adversarial examples were discovered and formulated by Szegedy et al. [219], and Goodfellow et al. [83]. They find that by adding a small but specifically designed perturbation to an image, one can make the prediction of an accurate machine learning model incorrect (see Figure 1.1). Following this discovery, different ways of generating adversarial examples for different applications are being proposed. These applications include natural language processing [247], reinforcement learning [17, 18], and speech recognition [174]. The existence of these adversarial examples can cause significant safety issues, and there are many attack algorithms that are capable of compromising the performance of real-world machine learning models [66, 218].

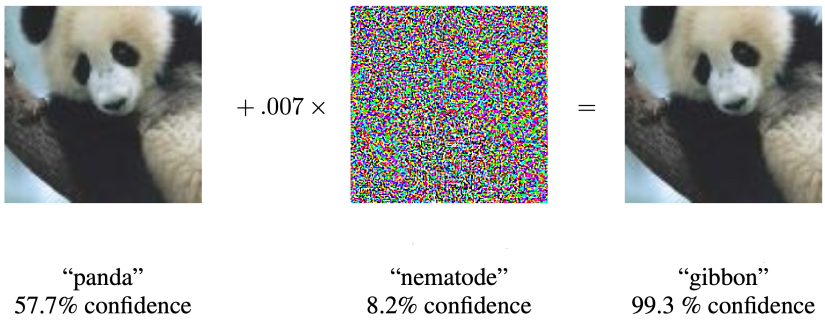


Figure 1.1: An example of adversarial example from Goodfellow et al. [83].

With these vulnerabilities being discovered, many defense algorithms are being proposed [52, 144, 190, 244]. However, one thing these defense algorithms have in common is that they are known to hurt the clean test accuracies [144, 177, 248]. This observation has led prior work to claim that tradeoffs between robustness and accuracy may be inevitable for many classification tasks [222, 244].

In Chapter 3, we look at neural networks and investigate the research question – *is the*

tradeoff between robustness and accuracy an intrinsic property? To answer the question, we first show that if the data is *r-separated* – meaning that all pairs of differently labeled examples are at least $2r$ away from each other, there exists an accurate and robust (to perturbations with a norm less than r) classifier. Next, we find that many image datasets are r-separated. This implies that it is possible to achieve both robustness and accuracy at the same time for these image classification problems. While the theory suggests there is no intrinsic tradeoff between robustness and accuracy on many image datasets, in practice, many works report that there is a tradeoff that is unable to mitigate [222, 244]. With extensive experiments, our results suggest that this gap between theory and practice is caused by the fact that many robust learning algorithms can compromise the generalization ability of neural networks. This implies that future work should focus on the generalization issue of adversarial robust models.

In Chapter 4, we focus on non-parametric classifiers, including k-nearest neighbors and decision trees, and develop an algorithm to make them robust to adversarial examples. Unlike neural networks, non-parametric classifiers often operate on tabular datasets, and these datasets are commonly not r-separated, meaning that the distance between differently labeled examples can be $< 2r$. We start by observing that when the distance between two differently labeled examples is $< 2r$ (i.e., not r-separated), one of these two examples must be either incorrectly predicted or non-robust. Based on this observation, we design our defense algorithm – adversarial pruning, which reprocesses and removes the minimum number of examples so that the dataset becomes r-separated. To evaluate our defense, we also develop a novel attack algorithm – region-based attack, which achieves optimal attack and applies to a wide range of non-parametric classifiers. Through extensive experiments, we show that our defense and attack are better than or competitive with prior work. Overall, our results provide a strong and broadly-applicable baseline for future work.

In Chapter 5, we investigate the connection between accuracy and robustness with another property – interpretability. This question can be hard to answer for general classifiers; therefore,

we focus on the decision tree. We begin the analysis with the assumption that the data is r-separated as in Chapter 3, we have shown that r-separation is a necessary condition for a classifier to be accurate and robust. Our analysis shows that, in the worst case, the depth of the decision tree can grow exponentially in the dimension of the feature if we want an accurate tree. This result indicates that having a r-separated data cannot guarantee that there exists a tree that is accurate and interpretable. Next, we shift our focus to a stricter condition – γ-linearly separable data (where γ is the margin of the linearly separable data). Under this assumption, we can construct a decision tree that is not only robust and accurate, but the depth of the tree is upper-bounded by $O(\gamma^{-2}\log(1/\varepsilon))$. This means that the tree is also interpretable as the depth is independent of the number of samples and feature dimensions. Finally, with extensive experiments, our algorithm provides better interpretability and robustness while having a comparable accuracy compared to other tree-based methods.

1.2 Out-of-Distribution Generalization

Out-of-distribution (OOD) generalization studies how classifiers behave when an example that does not belong to the training set is given as an input. There are various learning problems related to OOD generalization, including transfer learning [191], outlier detection [147], and few-shot learning [116]. In Chapter 6, we investigate the question *"is there any pattern in a neural network's output when natural OOD examples are given?"* Motivated by a line of work in the psychology literature which posits that humans categorize unseen examples into the most similar category they have seen before [12, 159, 185, 192]. For example, when a child sees an orange for the first time, he may categorize an orange as a type of similar fruit he has seen before, such as a tangerine. Inspired by this unique tendency of humans, we investigate whether neural networks show similar behavior.

We test whether neural networks also tend to predict OOD examples as the nearest

category in the training set, and we call this property Nearest Category Generalization (NCG). We begin with setting up a framework for testing whether neural networks show signs of NCG. We take existing datasets, remove examples of a specific class from the training set, and treat these removed examples as the OOD examples. We find that across forty different neural networks trained across four different datasets, **all** neural networks show a significant NCG behavior. In addition, we also repeat the experiment on networks trained to be robust, and we find the NCG behavior of these networks is amplified.

Prior work observes that robust networks can perform better in many OOD-related tasks such as transfer learning and corrupted data [191, 225]; however, why this happens is still unknown. We start by asking the question *why does robust training amplifies the NCG behavior?* A plausible explanation is those robust training algorithms like TRADES [244] enforce the network to be locally smooth in a ball of radius r around training data [240]; if the OOD inputs are closer than r from their nearest training example, then they would get classified in the same class. Surprisingly, we find that this is not the case. Balls of radius r around most training examples are so small that they cover almost none of the OOD inputs. Moreover, OOD inputs that are classified with their nearest categories are considerably further from their closest training examples, which, in turn, continue to have adversarial examples that are closer than the robustness radius r (see Figure 1.2). This suggests that the robust neural networks may be smoother in some directions than others and perhaps smoother than they were trained to be along the natural image manifold. As images of unseen classes should also lie on the natural image manifold, this observation sheds light on why robust networks are able to generalize better to OOD data.

All in all, we posit that the NCG property is a consequence of the inductive bias produced by neural networks (especially for adversarially robust networks). It is interesting that this inductive bias happens to be similar to some human behaviors, and enforcing adversarial robustness, which is another feature that humans possess, can make the NCG property more salient. Many scholars conjecture that the effectiveness of deep learning may be coming from its

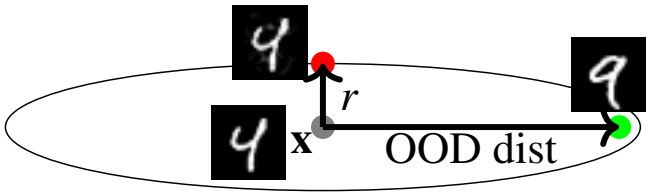


Figure 1.2: Robust networks tend to predict smoothly at a larger distance in some directions, e.g., toward natural OOD examples (green point), but are susceptible to adversarial examples that are closer in the worst-case directions (red point).

similar structure to the human brain [91, 127, 198], which allows the neural networks to share some of the inductive biases from the brain. This work can be an additional piece of evidence supporting this theory. In addition, how neural networks generalize so well is still an open question [137]. Our work provides some insights into how neural networks generalize.

1.3 Rare Spurious Correlation

Using spurious correlations to make predictions can also make a classifier fail at test time. Neural networks are known to spuriously correlate certain features with certain classes [89, 189, 213, 249]. For example, Sagawa et al. [189] show that models trained on the Waterbirds dataset [188] correlate waterbirds with backgrounds containing water, and models trained on the CelebA dataset [139] correlate males with dark hair. This can cause a significant detriment to generalization when there is a distributional shift in test data [155, 188]. In all these cases, the spurious patterns are presented in a substantial number of the training data. For instance, in the Waterbirds dataset, the vast majority of waterbirds are photographed next to the water.

In Chapter 7, we ask the question we ask here is *whether rare spurious patterns that only occur in a handful of examples are also learned by neural networks.* If yes, then even a small number of examples could negatively affect OOD generalization; additionally, the rarity of these examples may pose a potential privacy concern. As an illustration, consider the example in Leino and Fredrikson [131], where the training set had an image of Tony Blair with a pink background.

This led to a classifier that assigned a higher likelihood of the label "Tony Blair" to all images with pink backgrounds. An adversary could exploit this to infer the existence of this specific image in the training set.

To answer our research question, we introduce spurious correlations into real image datasets by adding different spurious patterns into a number of training images belonging to a target class. These are the spurious examples. We then train a neural network on the modified dataset and measure the strength of the correlation between the spurious pattern and the target class in the network. We find that even a network trained with **just** 3 **spurious examples**, this correlation can be significantly higher over the baseline; additionally, visualizations show that the network's weights may also be significantly affected. Therefore, rare spurious correlations that occur in a small number of training examples can be learned by neural networks.

Recent privacy law such as GDPR allows individuals to request the removal of their data, and this includes removing the data from the models that have been trained on these data. If all spurious examples are deleted from the model, then the model should not have learned the spurious correlation. We take two commonly used data deletion methods – incremental retraining and influence functions [15, 117], and examine whether the spurious correlations can be removed by using them to delete the spurious examples. We find that the spurious correlations remain after the spurious examples are deleted using these methods.

In summary, our findings suggest that neural networks can learn spurious correlations even if only a small number of spurious examples are in the training set. This brings up several significant concerns about the use of deep learning in societal applications, such as privacy [131] and fairness [103] problems. Our results also reveal that some data deletion methods may not remove spurious correlations introduced by the deleted examples. This motivates the development of better data deletion algorithms with performance guarantees.

1.4 Overview

This thesis is organized as follows.

- Chapter 2 goes through the background material on adversarial examples and setup common notations.
- Chapter 3 presents our study on the tradeoff between accuracy and adversarial robustness. This joint work with Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri has been published in NeurIPS 2020 [240].
- Chapter 4 shows our work on the adversarial attack and defense algorithms for non-parametric classifiers. This joint work with Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri has been published in AISTATS 2020 [239].
- Chapter 5 investigates the connection of adversarial robustness, accuracy, and interpretability for decision trees. This joint work with Michal Moshkovitz and Kamalika Chaudhuri has been published in ICML 2021 [152].
- Chapter 6 explores the output of neural networks with out-of-distribution input. This joint work with Cyrus Rashtchian, Ruslan Salakhutdinov, and Kamalika Chaudhuri [238] is in submission.
- Chapter 7 contains our research on the phenomena of rare spurious correlation. This joint work with Kamalika Chaudhuri [235] is in submission.
- Chapter 8 concludes this thesis. This thesis aims to systematically examine the behavior of a machine learning model when training data and testing data are not drawn independently from the same distribution.

Chapter 2

Adversarial Robustness Basics

In this chapter, we setup the notations and basics the adversarial robustness setup in Chapters 3 to 5.

2.1 Multi-Class Classification

We consider adversarial robustness under the multi-class classification setting. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be an instance space equipped with a metric $\mathsf{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. At training time, we are given a set of examples $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, where each $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ is associated with a label $y_i \in [C]$, where $[C] = \{1, \ldots, C\}$. A classifier uses the training data to learn a function $f : \mathcal{X} \to [C]$, that maps each example to a label. At test time, we evaluate the network on test data examples drawn independently from the training distribution. We compute the *clean test accuracy* as $\Pr_{(\mathbf{x},y)\sim\mu}[f(\mathbf{x}) = y]$.

2.2 Adversarial Robustness

Adversarial attack algorithm. An adversarial attack algorithm A takes in the target

classifier f, the target example $\mathbf{x}$, and the attack radius r and outputs an adversarial example. There are two main threat models have been proposed. The *black-box* setting restricts the adversary to only querying a classifier f on various inputs. Some commonly seen black-box attack algorithms include transfer attack [167], BBox [50], and Sign-OPT [51]. In the *white-box* setting, the adversary has full access to f, including the model structure and parameters. Some commonly seen white-box attack algorithms include projected gradient descent (PGD) [144], and C&W [39].

Adversarial example. The adversary's goal is to modify a true input by a small amount and cause the classifier to output the wrong label. Fix a classifier f, a norm $\|\cdot\|$ on the instance space $\mathbb{R}^d$ and distance metric dist. An *adversarial example* for f at $\mathbf{x}$ is any other input $\mathbf{x}'$ such that $\|\mathbf{x}-\mathbf{x}'\| < r$ and $f(\mathbf{x}) \neq f(\mathbf{x}')$ for some small constant r.

We define the optimal adversarial example as the adversarial example $\mathbf{x}'$ that is the closest vector to $\mathbf{x}$ that receives a different label.

Definition 1 (Optimal adversarial example)**.** *An* optimal adversarial example *for f at $\mathbf{x}$ is an input $\mathbf{x}'$ that minimizes $\|\mathbf{x}-\mathbf{x}'\|$ subject to $f(\mathbf{x}) \neq f(\mathbf{x}')$.*

In practice, it is not always possible to find the optimal adversarial example, and hence the goal is to find $\mathbf{x}'$ that is as close to $\mathbf{x}$ as possible.

2.2.1 Evaluation Criteria

Robustness and astuteness. Let $\mathbb{B}(\mathbf{x}, r)$ denote a ball of radius $r > 0$ around $\mathbf{x}$ in a metric space. We use $\mathbb{B}_\infty$ to denote the ℓ_∞ ball. A classifier g is *robust* at $\mathbf{x}$ with radius $r > 0$ if for all $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, r)$, we have $f(\mathbf{x}') = f(\mathbf{x})$. Also, f is *astute* at $(\mathbf{x}, y)$ if $f(\mathbf{x}') = y$ for all $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, r)$.

Definition 2 (Astuteness $\mathsf{ast}_\mu(f, r)$)**.** *The* astuteness *of f at radius $r > 0$ under a distribution μ is*

$$\Pr_{(\mathbf{x},y)\sim\mu}[f(\mathbf{x}') = y \textit{ for all } \mathbf{x}' \in \mathbb{B}(\mathbf{x}, r)].$$

Astuteness measures the accuracy in the worst-case scenario where adversaries are always able to find the optimal adversarial example. However, in many cases, finding the optimal adversarial example is infeasible. We define the empirical astuteness in Definition 3.

Definition 3 (Empirical astuteness (adversarial accuracy) $\mathsf{ast}_\mu(A, f, r)$)**.** *Astuteness is the percentage of correctly predicted examples after adversarial attack. Let μ be the distribution that the training data is sampled from.*

$$\mathsf{ast}_\mu(A, f, r) := \Pr_{(\mathbf{x},y)\sim\mu}[f(A(f, \mathbf{x}, r)) = y \text{ and } f(\mathbf{x}) = y].$$

We use *clean accuracy* to refer to standard test accuracy (no adversarial perturbation) to differentiate it from *adversarial accuracy* a.k.a. empirical astuteness (with adversarial perturbation).

Robustness radius. We also define the robustness radius, which is the minimum perturbation needed to change the classifier label.

Definition 4 (Robust radius $\rho(f, \mathbf{x})$)**.** *Let $\mathcal{X} \times [C]$ be a labeled space with norm $\|\cdot\|$. The* robustness radius *of f at $\mathbf{x} \in \mathcal{X}$ is*

$$\rho(f, \mathbf{x}) := \min_{\mathbf{x}' \in \mathcal{X}} \{\|\mathbf{x} - \mathbf{x}'\| : f(\mathbf{x}) \neq f(\mathbf{x}')\}.$$

Empirical robustness. Robust radius considers the minimum perturbation, which is hard to measure for arbitrary f in practice. Thus, we define the empirical version of the measurement for robust radius as empirical robustness (ER). ER measures the distance of $\mathbf{x}$ to the adversarial example found by the attack algorithm A.

Definition 5 (Empirical Robustness ($\mathsf{ER}(A, f, \mathbf{x})$))**.** *Let the input be attack algorithm A on target classifier f at an example set S with t examples.*

$$\begin{aligned} \mathsf{ER}(A, f, \mathbf{x}) &:= \|\mathbf{x} - A(f, \mathbf{x}, \infty)\| \\ \mathsf{ER}(A, f, S, t) &:= \frac{1}{t} \sum_{\mathbf{x} \in S} \mathsf{ER}(A, f, \mathbf{x}) \end{aligned} \tag{2.1}$$

Evaluation on attacks. A reasonable method to evaluate the performance of an adversarial attack algorithm is to measure the empirical astuteness [144]. With a fixed classifier f, the lower empirical astuteness is, the stronger an attack algorithm is. Another commonly used metric is the empirical robustness (ER) [47, 50, 108]. With a fixed classifier f, the smaller the ER is, the stronger an attack algorithm is.

Empirical astuteness measures the accuracy after the attack, while empirical robustness measures the ability of a classifier to withstand small changes. Empirical astuteness is a more natural way of measuring the robustness of a model. However, different classifiers can have very different baseline accuracy. When comparing models that are very different, using empirical astuteness may not reflect the actual robustness. Thus, it is more common to use empirical robustness in this case.

Adversarial defense algorithms. The goal of an adversarial robust algorithm (or a defense algorithm) is to find a f with a high astuteness [228]. We denote f_D as the classifier trained with defense algorithm D. In practice, the success of an adversarial robust algorithms is usually measured with empirical astuteness (adversarial accuracy). Some commonly seen defense algorithms include adversarial training [144], TRADES [244], and robust trees [47].

Evaluation on defenses. We can also evaluate a defense algorithm D by measuring how much ER increased after the defense, and we call this measurement the defscore. The *defscore* with respect to an attack A, a test set S and test size t is the ratio where f is the undefended classifier. A larger defscore implies a better defense.

Definition 6 (defscore(D, A, f, S, t))**.** *Let the input be defense algorithm D, attack algorithm A,*

undefended target classifier f, and testing set S with t examples. f_D is the classifier trained with defense algorithm D.

$$defscore(D,A,f,S,t) = \frac{\mathsf{ER}(A,f_D,S,t)}{\mathsf{ER}(A,f,S,t)},$$

Chapter 3

Adversarial Robustness for Neural Networks

3.1 Introduction

A growing body of research shows that neural networks are vulnerable to *adversarial examples*, test inputs that have been modified slightly yet strategically to cause misclassification [83, 219]. While a number of defenses have been proposed [52, 144, 190, 244], they are known to hurt test accuracy on many datasets [144, 177, 248]. This observation has led prior works to claim that a tradeoff between robustness and accuracy may be *inevitable* for many classification tasks [222, 244].

We take a closer look at the tradeoff between robustness and accuracy, aiming to identify properties of data and training methods that enable neural networks to achieve *both*. A plausible reason why robustness may lead to lower accuracy is that different classes are very close together or they may even overlap (which underlies the argument for an inevitable tradeoff [222]). We begin by testing if this is the case in real data through an empirical study of four image datasets. Perhaps surprisingly, we find that these datasets actually satisfy a natural separation property

that we call r-separation: examples from different classes are at least distance $2r$ apart in pixel space. This r-separation holds for values of r that are higher than the perturbation radii used in adversarial example experiments.

We next consider separation as a guiding principle for better understanding the robustness-accuracy tradeoff. Neural network classifiers are typically obtained by rounding an underlying continuous function $f : \mathcal{X} \to \mathbb{R}^C$ with C classes. We take inspiration from prior work, which shows that Lipschitzness of f is closely related to its robustness [52, 94, 190, 229, 243]. However, one drawback of the existing arguments is that they do not provide a compelling and realistic assumption on the data that guarantees robustness and accuracy. We show theoretically that any r-separated data distribution has a classifier that is both robust up to perturbations of size r, and accurate, and it can be obtained by rounding a function that is locally Lipschitz around the data. This suggests that there should exist a robust and highly accurate classifier for real image data. Unfortunately, the current state of robust classification falls short of this prediction, and the discrepancy remains poorly understood.

To better understand the theory-practice gap, we empirically investigate several existing methods on a few image datasets with a special focus on their local Lipschitzness and generalization gaps. We find that of the methods investigated, adversarial training (AT) [144], robust self-training (RST) [178] and TRADES [243] impose the highest degree of local smoothness, and are the most robust. We also find that the three robust methods have large gaps between training and test accuracies as well as adversarial training and test accuracies. This suggests that the disparity between theory and practice may be due to the limitations of existing training procedures, particularly in regards to generalization. We then experiment with dropout, a standard generalization technique, on top of robust training methods on two image datasets where there is a significant generalization gap. We see that dropout in particular narrows the generalization gaps of TRADES and RST, and improves test accuracy, test adversarial accuracy as well as test Lipschitzness. In summary, our contributions are as follows.

- Through empirical measurements, we show that several image datasets are separated.
- We prove that this separation implies the existence of a robust and perfectly accurate classifier that can be obtained by rounding a locally Lipschitz function. In contrast to prior conjectures [67, 81, 222], robustness and accuracy can be achieved together in principle.
- We investigate smoothness and generalization properties of classifiers produced by current training methods. We observe that the training methods AT, TRADES, and RST, which produce robust classifiers, also suffer from large generalization gaps. We combine these robust training methods with dropout [214], and show that this narrows the generalization gaps and sometimes makes the classifiers smoother.

What do our results imply about the robustness-accuracy tradeoff in deep learning? They suggest that this tradeoff is not inherent. Rather, it is a consequence of current robustness methods. The past few years of research in robust machine learning has led to a number of new loss functions, yet the rest of the training process – network topologies, optimization methods, generalization tools – remain highly tailored to promoting accuracy. We believe that in order to achieve both robustness and accuracy, future work may need to redesign other aspects of the training process such as better network architectures using neural architecture search [70, 87, 194, 250]. Combining this with improved optimization methods and robust losses may be able to reduce the generalization gap in practice.

3.2 Preliminaries

We follow the multi-class classification and adversarial robustness setup in Section 2.1.

Local Lipschitzness. Here we define local Lipschitzness theoretically; Section 3.5 later provides an empirical way to estimate this quantity.

Definition 7 (L-locally Lipschitz)**.** *Let $(\mathcal{X}, \mathsf{dist})$ be a metric space. A function $f : \mathcal{X} \to \mathbb{R}^C$ is L-locally Lipschitz at radius r if for each $i \in [C]$, we have $|f(\mathbf{x})_i - f(\mathbf{x}')_i| \leq L \cdot \mathsf{dist}(\mathbf{x}, \mathbf{x}')$ for all $\mathbf{x}'$ with $\mathsf{dist}(\mathbf{x}, \mathbf{x}') \leq r$.*

Separation. We formally define separated data distributions as follows. Let $\mathcal{X}$ contain C disjoint classes $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(C)}$, where all points in $\mathcal{X}^{(i)}$ have label i for $i \in [C]$.

Definition 8 (r-separation)**.** *We say that a data distribution over $\bigcup_{i \in [C]} \mathcal{X}^{(i)}$ is r-separated if $\mathsf{dist}(\mathcal{X}^{(i)}, \mathcal{X}^{(j)}) \geq 2r$ for all $i \neq j$, where $\mathsf{dist}(\mathcal{X}^{(i)}, \mathcal{X}^{(j)}) = \min_{\mathbf{x} \in \mathcal{X}^{(i)}, \mathbf{x}' \in \mathcal{X}^{(j)}} \mathsf{dist}(\mathbf{x}, \mathbf{x}')$.*

In other words, the distance between any two examples from different classes is at least $2r$. One of our motivating observations is that many real classification tasks comprise of separated classes; for example, if dist is the ℓ_∞ norm, then images with different categories (e.g., dog, cat, panda, etc) will be r-separated for some value $r > 0$ depending on the image space. In the next section, we empirically verify that this property actually holds for a number of standard image datasets.

3.3 Real Image Datasets are r-Separated

We begin by addressing the question: Are image datasets r-separated for $\varepsilon \ll r$ and attack radii ε in standard robustness experiments? While we cannot determine the underlying data distribution, we can empirically measure whether current training and test sets are r-separated. These measurements can potentially throw light on what can be achieved in terms of test robustness in real data.

We consider four datasets: MNIST, CIFAR-10, SVHN and Restricted ImageNet (ResImageNet), where ResImageNet contains images from a subset of ImageNet classes [144, 184, 222]. We present two statistics in Table 3.1 The *Train-Train Separation* is the ℓ_∞ distance between each training example and its closest neighbor with a different class label in the training set, while the

Test-Train Separation is the ℓ_∞ distance between each test example and its closest example with a different class in the training set. See Figure 3.2 for histograms. We use exact nearest neighbor search to calculate distances. Table 3.1 also shows the typical adversarial attack radius ε for the datasets; more details are presented in the Appendix B.3.

Both the Train-Train and Test-Train separations are higher than 2ε for all four datasets. We note that SVHN contains a single duplicate example with multiple labels, and one highly noisy example; removing these three examples out of $73,257$ gives us a minimum Train-Train Separation of 0.094, which is more than enough for attack radius $\varepsilon = 0.031 \approx 8/255$. Restricted ImageNet is similar with three pairs of duplicate examples, and two other highly noisy training examples (see Figures B.2 and B.3 in Appendix B.3). Barring a handful of highly noisy examples, real image datasets are indeed r-separated when r is equal to the attack radii commonly used in adversarial robustness experiments.

Table 3.1: Separation of real data is $3\times$ to $7\times$ typical perturbation radii.

	adversarial perturbation ε	minimum Train-Train separation	minimum Test-Train separation
MNIST	0.1	0.737	0.812
CIFAR-10	0.031	0.212	0.220
SVHN	0.031	0.094	0.110
ResImageNet	0.005	0.180	0.224

These results imply that in real image data, the test images are far apart from training images from a different class. There perhaps are images of dogs which look like cats, but standard image datasets are quite clean, and such images mostly do not occur in either their test nor the training sets. In the next section, we explore consequences of this separation.

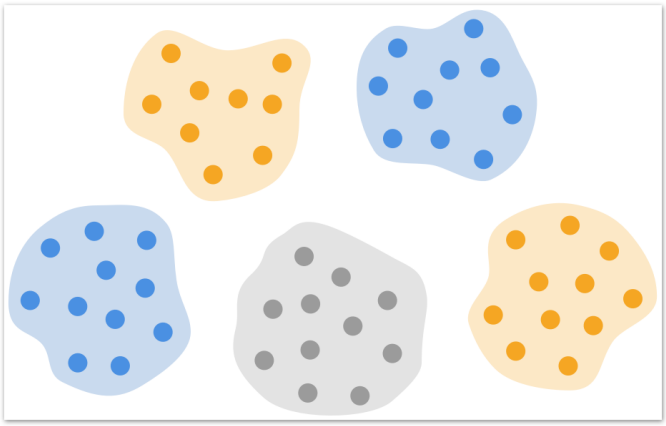
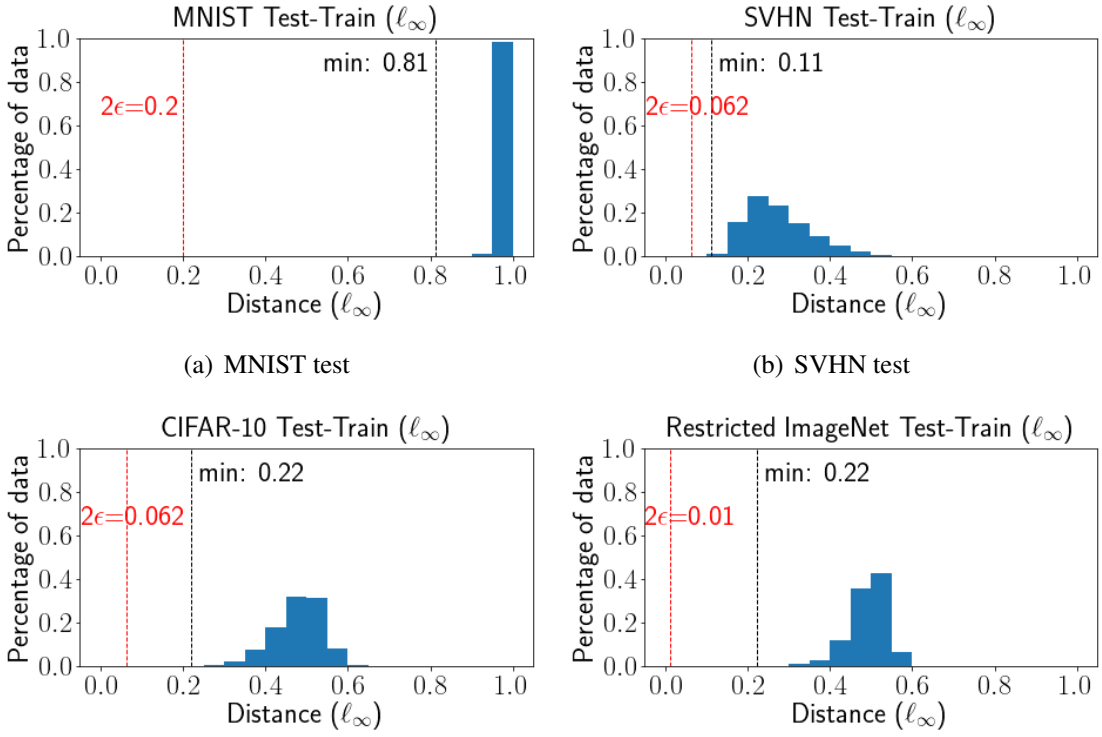


Figure 3.1: Robust classifiers exist if the perturbation is less than the separation.



(a) MNIST test

(b) SVHN test

(c) CIFAR-10 test

(d) Restricted ImageNet test

Figure 3.2: Train-Test separation histograms: MNIST, SVHN, CIFAR-10 and Restricted ImageNet.

3.4 Robustness and Accuracy for r-Separated Data

We have just shown that four image datasets are indeed r-separated, for $\varepsilon \ll r$ where ε is the typical adversarial perturbation used in experiments. We now show theoretically that if a data distribution is r-separated, then there exists a robust and accurate classifier that can be obtained by rounding a locally Lipschitz function. Additionally, we supplement these results in Appendix B.2 by a constructive "existence proof" that demonstrates proof-of-concept neural networks with both high accuracy and robustness on some of these datasets; this illustrates that at least on these image datasets, these classifiers can potentially be achieved by neural networks.

3.4.1 r-Separation Implies Robustness and Accuracy through Local Lipschitzness

We show that it is theoretically possible to achieve both robustness and accuracy for r-separated data. In particular, we exhibit a classifier based on a locally Lipschitz function, which has astuteness (Definition 2) 1 with radius r. Working directly in the multiclass case, our proof uses classifiers of the following form. If there are C classes, we start with a vector-valued function $f : \mathcal{X} \to \mathbb{R}^C$ so that $f(\mathbf{x})$ is a C-dimensional real vector. We let $\mathsf{dist}(\mathbf{x}, \mathcal{X}^{(i)}) = \min_{\mathbf{z} \in \mathcal{X}^{(i)}} \mathsf{dist}(\mathbf{x}, \mathbf{z})$. We analyze the following function

$$f(\mathbf{x}) = \frac{1}{r} \cdot \left(\mathsf{dist}(\mathbf{x}, \mathcal{X}^{(1)}), \ldots, \mathsf{dist}(\mathbf{x}, \mathcal{X}^{(C)}) \right), \tag{3.1}$$

In other words, we set $f(\mathbf{x})_i = \frac{1}{r} \cdot \mathsf{dist}(\mathbf{x}, \mathcal{X}^{(i)})$. Then, we define a classifier $g : \mathcal{X} \to [C]$ as

$$g(\mathbf{x}) = \operatorname*{argmin}_{i \in [C]} f(\mathbf{x})_i. \tag{3.2}$$

We show that accuracy and local Lipschitzness together imply astuteness.

Lemma 3.4.1. *Let $f : \mathcal{X} \to \mathbb{R}^C$ be a function, and consider $\mathbf{x} \in \mathcal{X}$ with true label $y \in [C]$. If*

- *f is $\frac{1}{r}$-Locally Lipschitz in a radius r around $\mathbf{x}$, and*
- *$f(\mathbf{x})_j - f(\mathbf{x})_y \geq 2$ for all $j \neq y$,*

then $g(\mathbf{x}) = \operatorname{argmin}_i f(\mathbf{x})_i$ is astute at $\mathbf{x}$ with radius r.

Proof. Suppose $\mathbf{x}' \in \mathcal{X}$ satisfies $\mathsf{dist}(\mathbf{x}, \mathbf{x}') \leq r$. By the assumptions that f is $\frac{1}{r}$-locally Lipschitz and $f(\mathbf{x})_j - f(\mathbf{x})_y \geq 2$, we have that

$$f(\mathbf{x}')_j \geq f(\mathbf{x})_j - 1 \geq f(\mathbf{x})_y + 1 \geq f(\mathbf{x}')_y,$$

where the first and third inequalities use Lipschitzness, and the middle inequality uses that

$$f(\mathbf{x})_j - f(\mathbf{x})_y \geq 2.$$

As this holds for all $j \neq y$, we have that

$$\operatorname*{argmin}_i f(\mathbf{x}')_i = \operatorname*{argmin}_i f(\mathbf{x})_i = y.$$

Thus, we see that $g(\mathbf{x}) = \operatorname{argmin}_i f(\mathbf{x})_i$ correctly classifies $\mathbf{x}$ while being astute with radius r. □

Finally, we show that there exists an astute classifier when the distribution is r-separated.

Theorem 3.4.2. *Suppose the data distribution $\mathcal{X}$ is r-separated, denoting C classes $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(C)}$. There exists a function $f : \mathcal{X} \to \mathbb{R}^C$ such that*

(a) f is $\frac{1}{r}$-locally-Lipschitz in a ball of radius r around each $\mathbf{x} \in \bigcup_{i \in [C]} \mathcal{X}^{(i)}$, and

(b) the classifier $g(\mathbf{x}) = \operatorname{argmin}_i f(\mathbf{x})_i$ has astuteness 1 with radius r.

Proof. We first show that if the support of the distribution is r-separated, then there exists a function $f : \mathcal{X} \to \mathbb{R}^C$ satisfying:

1. If $\mathbf{x} \in \bigcup_{i \in [C]} \mathcal{X}^{(i)}$, then, $f(\mathbf{x})$ is $\frac{1}{r}$-locally-Lipschitz in a ball of radius r around $\mathbf{x}$.

2. If $\mathbf{x} \in \mathcal{X}^{(y)}$, then $f(\mathbf{x})_j - f(\mathbf{x})_y \geq 2$ for all $j \neq y$.

Define the function

$$f(\mathbf{x}) = \frac{1}{r} \cdot \left(\mathsf{dist}(\mathbf{x}, \mathcal{X}^{(1)}), \ldots, \mathsf{dist}(\mathbf{x}, \mathcal{X}^{(C)}) \right).$$

In other words, we set $f(\mathbf{x})_i = \frac{1}{r} \cdot \mathsf{dist}(\mathbf{x}, \mathcal{X}^{(i)})$. Then, for any $\mathbf{x}$, we have:

$$f(\mathbf{x})_i - f(\mathbf{x}')_i = \frac{\mathsf{dist}(\mathbf{x}, \mathcal{X}^{(i)}) - \mathsf{dist}(\mathbf{x}', \mathcal{X}^{(i)})}{r} \leq \frac{\mathsf{dist}(\mathbf{x}, \mathbf{x}')}{r}$$

where we used the triangle inequality. This establishes (a). To establish (b), suppose without loss of generality that $\mathbf{x} \in \mathcal{X}^{(y)}$, which in particular implies that $f(\mathbf{x})_y = \mathsf{dist}(\mathbf{x}, \mathcal{X}^{(y)}) = 0$. Then,

$$f(\mathbf{x})_j - f(\mathbf{x})_y = \frac{\mathsf{dist}(\mathbf{x}, \mathcal{X}^{(j)})}{r} \geq \frac{\mathsf{dist}(\mathcal{X}^{(y)}, \mathcal{X}^{(j)})}{r} \geq 2$$

because every pair of classes has distance at least $2r$ from the r-separated property.

Now observe that by construction, f satisfies the two conditions in Lemma 3.4.1 at all $\mathbf{x} \in \bigcup_{i \in [C]} \mathcal{X}^{(i)}$. Thus, applying Lemma 3.4.1, we get that $g(\mathbf{x}) = \operatorname{argmin}_i f(\mathbf{x})_i$ has astuteness 1 with radius r over any distribution over points in $\bigcup_{i \in [C]} \mathcal{X}^{(i)}$. □

While the classifier g used in the proof of Theorem 3.4.2 resembles the 1-nearest-neighbor classifier, it is actually different on any finite sample, and the classifiers only coincide in the *infinite sample limit* or when the class supports are known.

Binary case. We also state results for the special case of binary classification. Let $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^-$ be the instance space with disjoint class supports $\mathcal{X}^+ \cap \mathcal{X}^- = \emptyset$. Then, we define $f : \mathcal{X} \to \mathbb{R}$ as

$$f(\mathbf{x}) = \frac{\mathsf{dist}(\mathbf{x}, \mathcal{X}^-) - \mathsf{dist}(\mathbf{x}, \mathcal{X}^+)}{2r}.$$

It is immediate that if $\mathcal{X}$ is r-separated, then f is locally Lipschitz with constant $1/r$, and also the

classifier $g = \mathsf{sign}(f)$ achieves the guarantees in the following theorem using the next lemma.

Lemma 3.4.3. *Let $f : \mathcal{X} \to \mathbb{R}$, and let $\mathbf{x} \in \mathcal{X}$ have label y. If (a) f is $\frac{1}{r}$-Locally Lipschitz in a ball of radius $r > 0$ around $\mathbf{x}$, (b) $|f(\mathbf{x})| \geq 1$, and (c) $g(\mathbf{x})$ has the same sign as y, then the classifier $g = \mathsf{sign}(f)$ is astute at $\mathbf{x}$ with radius r.*

Theorem 3.4.4. *Suppose the data distribution $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^-$ is r-separated. Then, there exists a function f such that (a) f is $\frac{1}{r}$-locally Lipschitz in a ball of radius r around all $\mathbf{x} \in \mathcal{X}$ and (b) the classifier $g = \mathsf{sign}(f)$ has astuteness 1 with radius r.*

A visualization of the function (and resulting classifier) from Theorem 3.4.4 for a binary classification dataset appears in Figure 3.3. Dark colors indicate high confidence (far from decision boundary) and lighter colors indicate the gradual change from one label to the next. The classifier $g = \mathsf{sign}(f)$ guaranteed by this theorem will predict the label based on which decision region (positive or negative) is closer to the input example. Figure 3.4 shows a pictorial example of why using a locally Lipschitz function can be just as expressive while also being robust.

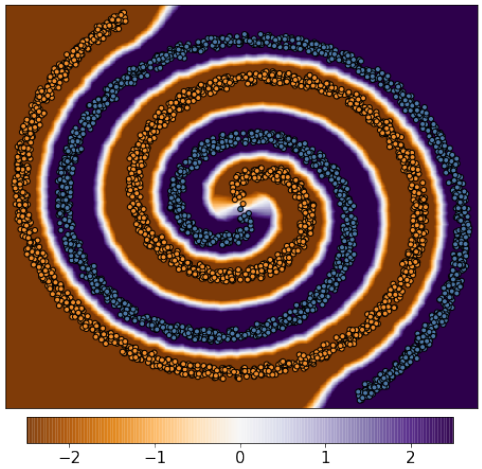


Figure 3.3: Plot of $f(\mathbf{x})$ from Theorem 3.4.4 for the spiral dataset. The classifier $g = \mathsf{sign}(f)$ has high accuracy and astuteness because it gradually changes near the decision boundary.

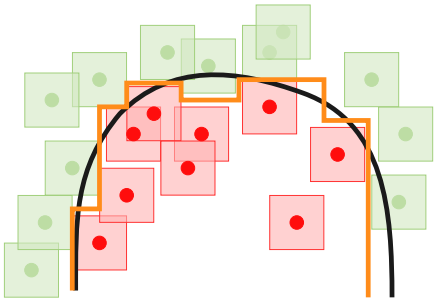


Figure 3.4: The classifier corresponding to the orange boundary has small local Lipschitzness because it does not change in the ℓ_∞ balls around data points. The black curve, however, is vulnerable to adversarial examples even though it has high clean accuracy.

3.4.2 Implications

We consider the consequences of Table 3.1 and Theorem 3.4.2 taken together. Then, in Section 3.5, we empirically explore the limitations of current robustness techniques.

Significance for real data. Theorem 3.4.2 refers to supports of distributions, while our measurements in Table 3.1 are on actual data. Hence, the results do not imply perfect *distributional* accuracy and robustness. However, our test set measurements suggest that even if the distributional supports may be close in the infinite sample limit, the close images are rare enough that we do not see them in the test sets. Thus, we still expect high accuracy and robustness on these test sets. Additionally, if we are willing to assume that the data supports are representative of the support of the distribution, then we can conclude the existence of a distributionally robust and accurate classifier. Combined with proof-of-concept results in Appendix B.2, we deduce that these classifiers can be implemented by neural networks. The remaining question is how such networks can be *trained* with existing methods.

Optimally astute classifier and non-parametrics (comparison to Yang et al. [239]). Prior work proposes adversarial pruning, a method that removes training examples until different classes are r-separated. They exhibit connections to maximally astute classifier, which they call the r-Optimal classifier for size r perturbations [239]. Follow-up work proved that training various non-parametric classifiers after pruning leads them to converge to maximally astute classifiers

under certain conditions [22]. Our result in Theorem 3.4.2 complements these efforts, showing that the r-Optimal classifier can be obtained by the classifier. Moreover, we provide additional justification for adversarial pruning by presenting a new perspective on the role of data separation in robustness.

Lower bounds on test error. Our results also corroborate some recent works that use optimal transport to estimate a lower bound on the robust test accuracy of any classifier on standard datasets. They find that it is actually zero for typical perturbation sizes [21, 171]. In other words, we have further evidence that well-curated benchmark datasets are insufficient to demonstrate a tradeoff between robustness and accuracy, in contrast to predictions of an inevitable tradeoff [60, 67, 81, 222].

Robustness is *not* inherently at odds with accuracy (comparison to Tsipras et al. [222]). Prior work provides a theoretical example of a data distribution where any classifier with high test accuracy must also have small adversarial accuracy under ℓ_∞ perturbations. Their theorem led the authors to posit that (i) accuracy and robustness may be unachievable together due their inherently opposing goals, and (ii) the training algorithm may not be at fault [222]. We provide an alternative view.

Their distribution is defined using the following sampling process: the first feature is the binary class label (flipped with a small probability), and the other $d-1$ features are sampled from one of two $(d-1)$-dimensional Gaussian distributions either with mean r or $-r$ depending on the true example label. While the means are separated with distance $2r$, their distribution is *not* r-separated due to the noise in the first feature combined with the infinite support of the Gaussians. Their lower bound is tight and only holds for ℓ_∞ perturbations ε satisfying $\varepsilon \geq 2r$. Our experiments in Section 3.3 have already shown that r-separation is a realistic assumption, and typical perturbations ε satisfy $\varepsilon \ll r$. Taken together with Theorem 3.4.2, we conclude that the robustness-accuracy tradeoff in neural networks and image classification tasks is *not intrinsic*.

3.5 A Closer Look at Existing Training Methods

So far we have shown that robustness and accuracy should both be achievable in principle, but practical networks continue to trade robustness off for accuracy. We next empirically investigate why this tradeoff might arise. One plausible reason might be that existing training methods do not impose local Lipschitzness properly; another may be that they do not generalize enough. We next explore these hypotheses in more detail, considering the following questions:

- How locally Lipschitz are the classifiers produced by existing training methods?
- How well do classifiers produced by existing training methods generalize?

These questions are considered in the context of one synthetic and four real datasets, as well as several plausible training methods for improving adversarial robustness. We do not aim to achieve best performance for any method, but rather to understand smoothness and generalization.

3.5.1 Experimental Methodology

We evaluate train/test accuracy, adversarial accuracy and local lipschitzness of neural networks trained using different methods. We also measure generalization gaps: the difference between train and test clean accuracy (or between train and test adversarial accuracy). The code for the experiments is available at `https://github.com/yangarbiter/robust-local-lipschitz`.

Training Methods. We consider neural networks trained via Natural training (Natural), Gradient Regularization (GR) [72], Locally Linear Regularization (LLR) [173], Adversarial Training (AT) [144], and TRADES [244]. Additionally, we use Robust Self Training (RST) [178], a recently introduced method that minimizes a linear combination of clean and adversarial accuracy in an attempt to improve robustness-accuracy tradeoffs. For fair comparison between methods, we use a version of RST that only uses labeled data. Both RST and TRADES have

a parameter; for RST higher λ means higher weight is given to the adversarial accuracy, while for TRADES higher β means higher weight given to enforcing local Lipschitzness. Details are provided in Appendix B.1.

Adversarial Attacks. We evaluate robustness with two attacks. In this section, we use Projected gradient descent (PGD) [124] for adversarial accuracy with step size $\varepsilon/5$ and a total of 10 steps. The Multi-Targeted Attack (MT) [86] leads to similar conclusions; results in Appendix B.4.1.

Measuring Local Lipschitzness. For each classifier, we empirically measure the local Lipschitzness of the underlying function by the *empirical Lipschitz constant* defined as the following quantity

$$\frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{x}_i'\in\mathbb{B}_\infty(\mathbf{x}_i,\varepsilon)}\frac{\|f(\mathbf{x}_i)-f(\mathbf{x}_i')\|_1}{\|\mathbf{x}_i-\mathbf{x}_i'\|_\infty}. \tag{3.3}$$

A lower value of the empirical Lipschitz constant implies a smoother classifier. We estimate this through a PGD-like procedure, where we iteratively take a step towards the gradient direction $(\nabla_{\mathbf{x}_i'}\frac{\|f(\mathbf{x}_i)-f(\mathbf{x}_i')\|_1}{\|\mathbf{x}_i-\mathbf{x}_i'\|_\infty})$ where ε is the perturbation radius. We use step size $\varepsilon/5$ and a total of 10 steps.

Datasets. We evaluate the various algorithms on one synthetic dataset: Staircase [177] and four real datasets: MNIST [130], SVHN [158], CIFAR-10 [121] and Restricted ImageNet [222]. We consider adversarial ℓ_∞ perturbations for all datasets. More details are in Appendix B.1.

3.5.2 Observations

Our experimental results, presented in Tables 3.2 and 3.3, provide a number of insights into the smoothness and generalization properties of classifiers trained by existing methods.

How well do existing methods impose local Lipschitzness? There is a large gap in the degree of local Lipschitzness in classifiers trained by AT, RST and TRADES and those trained by natural training, GR and LLR. Classifiers in the former group are considerably smoother than the latter. Classifiers produced by TRADES are the most locally Lipschitz overall, with smoothness

Table 3.2: MNIST (perturbation 0.1). We compare two networks: CNN1 (smaller) and CNN2 (larger). We evaluate adversarial accuracy with the PGD-10 attack and compute Lipschitzness with Equation (3.3). We also report the standard and adversarial generalization gaps.

architecture	CNN1						CNN2					
	train acc.	test acc.	adv test acc.	test lipschitz	gap	adv gap	train acc.	test acc.	adv test acc.	test lipschitz	gap	adv gap
Natural	100.00	99.20	59.83	67.25	0.80	0.45	100.00	99.51	86.01	23.06	0.49	-0.28
GR	99.99	99.29	91.03	26.05	0.70	3.49	99.99	99.55	93.71	20.26	0.44	2.55
LLR	100.00	99.43	92.14	30.44	0.57	4.42	100.00	99.57	95.13	9.75	0.43	2.28
AT	99.98	99.31	97.21	8.84	0.67	2.67	99.98	99.48	98.03	6.09	0.50	1.92
RST(λ=.5)	100.00	99.34	96.53	11.09	0.66	3.16	100.00	99.53	97.72	8.27	0.47	2.27
RST(λ=1)	100.00	99.31	96.96	11.31	0.69	2.95	100.00	99.55	98.27	6.26	0.45	1.73
RST(λ=2)	100.00	99.31	97.09	12.39	0.69	2.87	100.00	99.56	98.48	4.55	0.44	1.52
TRADES(β=1)	99.81	99.26	96.60	9.69	0.55	2.10	99.96	99.58	98.10	4.74	0.38	1.70
TRADES(β=3)	99.21	98.96	96.66	7.83	0.25	1.33	99.80	99.57	98.54	2.14	0.23	1.18
TRADES(β=6)	97.50	97.54	93.68	2.87	-0.04	0.37	99.61	99.59	98.73	1.36	0.02	0.80

Table 3.3: CIFAR-10 (perturbation 0.031) and Restricted ImageNet (perturbation 0.005). We evaluate adversarial accuracy with the PGD-10 attack and compute Lipschitzness with Equation (3.3).

	CIFAR-10						Restricted ImageNet					
	train acc.	test acc.	adv test acc.	test lipschitz	gap	adv gap	train acc.	test acc.	adv test acc.	test lipschitz	gap	adv gap
Natural	100.00	93.81	0.00	425.71	6.19	0.00	97.72	93.47	7.89	32228.51	4.25	-0.46
GR	94.90	80.74	21.32	28.53	14.16	3.94	91.12	88.51	62.14	886.75	2.61	0.19
LLR	100.00	91.44	22.05	94.68	8.56	4.50	98.76	93.44	52.62	4795.66	5.32	0.22
RST(λ=.5)	99.90	85.11	39.58	20.67	14.79	36.26	96.08	92.02	79.24	451.57	4.06	4.57
RST(λ=1)	99.86	84.61	40.89	23.15	15.25	41.31	95.66	92.06	79.69	355.43	3.61	4.67
RST(λ=2)	99.73	83.87	41.75	23.80	15.86	43.54	96.02	91.14	81.41	394.40	4.87	6.19
AT	99.84	83.51	43.51	26.23	16.33	49.94	96.22	90.33	82.25	287.97	5.90	8.23
TRADES(β=1)	99.76	84.96	43.66	28.01	14.80	44.60	97.39	92.27	79.90	2144.66	5.13	6.66
TRADES(β=3)	99.78	85.55	46.63	22.42	14.23	47.67	95.74	90.75	82.28	396.67	5.00	6.41
TRADES(β=6)	98.93	84.46	48.58	13.05	14.47	42.65	93.34	88.92	82.13	200.90	4.42	5.31

improving with increasing β. AT and RST also produce classifiers of comparable smoothness – but less smooth than TRADES. Overall, local Lipschitzness appears mostly correlated with adversarial accuracy; the more robust methods are also the ones that impose the highest degree of local Lipschitzness. But there are diminishing returns in the correlation between robustness *and* accuracy and local Lipschitzness; for example, the local smoothness of TRADES improves with higher β; but increasing β sometimes leads to drops in test accuracy even though the Lipschitz constant continues to decrease.

How well do existing methods generalize? We observe that for the methods that produce locally Lipschitz classifiers – namely, AT, TRADES and RST – also have large generalization gaps while natural training, GR and LLR generalize much better. In particular, there is a large gap between training and test accuracies of AT, RST and TRADES, and an even larger one between training and test adversarial accuracies. Although RST has better test accuracy than AT, it continues to have a large generalization gap with only labeled data. An interesting fact is that this generalization behavior is quite unlike linear classification, where imposing local Lipschitzness leads to higher margin and better generalization [234] – imposing local Lipschitzness in neural networks, at least through these methods, appears to hurt generalization instead of helping. This suggests that these robust training methods may not be generalizing properly.

3.5.3 A Closer Look at Generalization

A natural follow-up question is whether the generalization gap of existing methods can be reduced by existing generalization-promoting methods in deep learning. In particular, we ask: *Can we improve the generalization gap of AT, RST and TRADES through generalization tools?*

To better understand this question, we consider two medium-sized datasets, SVHN and CIFAR-10, which nevertheless have a reasonably high gap between the test accuracy of the model produced by natural training and the best robust model. We then experiment with dropout [214], a standard and highly effective generalization method. For SVHN, we use a dropout rate of 0.5 and for CIFAR-10 a rate of 0.2. More experimental details are provided in the Appendix B.1.

Table 3.4 shows the results, contrasted with standard training. We observe that dropout narrows the generalization gap between training and test accuracy, as well as adversarial training and test accuracy significantly for all methods. For SVHN, after incorporating dropout, the best test accuracy is achieved by RST (95.19%) along with an adversarial test accuracy of 55.22%; the best adversarial test accuracy (62.41%) is with TRADES ($\beta = 3$) along with a test accuracy of (94.10%). Both accuracies are much closer to the accuracy of natural training (96.66%), and the

Table 3.4: Dropout and generalization. SVHN (perturbation 0.031, dropout rate 0.5) and CIFAR-10 (perturbation 0.031, dropout rate 0.2). We evaluate adversarial accuracy with the PGD-10 attack and compute Lipschitzness with Equation (3.3).

		SVHN					CIFAR-10				
	dropout	test acc.	adv test acc.	test lipschitz	gap	adv gap	test acc.	adv test acc.	test lipschitz	gap	adv gap
Natural	False	95.85	2.66	149.82	4.15	0.87	93.81	0.00	425.71	6.19	0.00
Natural	True	96.66	1.52	152.38	3.34	1.22	93.87	0.00	384.48	6.13	0.00
AT	False	91.68	54.17	16.51	5.11	25.74	83.51	43.51	26.23	16.33	49.94
AT	True	93.05	57.90	11.68	-0.14	6.48	85.20	43.07	31.59	14.51	44.05
RST(λ=2)	False	92.39	51.39	23.17	6.86	36.02	83.87	41.75	23.80	15.86	43.54
RST(λ=2)	True	95.19	55.22	17.59	1.90	11.30	85.49	40.24	34.45	14.00	33.07
TRADES(β=3)	False	91.85	54.37	10.15	7.48	33.33	85.55	46.63	22.42	14.23	47.67
TRADES(β=3)	True	94.00	62.41	4.99	0.48	7.91	86.43	49.01	14.69	12.59	35.03
TRADES(β=6)	False	91.83	58.12	5.20	5.35	23.88	84.46	48.58	13.05	14.47	42.65
TRADES(β=6)	True	93.46	63.24	3.30	0.45	5.97	84.69	52.32	8.13	11.91	26.49

test adversarial accuracies are also significantly higher. A similar narrowing of the generalization gap is visible for CIFAR-10 as well. Dropout also appears to make the networks smoother as test Lipschitzness also appears to improve for all algorithms for SVHN, and for TRADES for CIFAR-10.

Dropout Improvements. Our results show that the generalization gap of AT, RST and TRADES can be reduced by adding dropout; this reduction is particularly effective for RST and TRADES. Dropout additionally decreases the test local Lipschitzness of all methods – and hence promotes generalization all round – in accuracy, adversarial accuracy, and also local Lipschitzness. This suggests that combining dropout with the robust methods may be a good strategy for overall generalization.

3.5.4 Implications

Our experimental results lead to three major observations. We see that the training methods that produce the smoothest and most robust classifiers are AT, RST and TRADES. However, these robust methods also do not generalize well, and the generalization gap narrows

when we add dropout.

Comparison with Rice et al. [182]. An important implication of our results is that generalization is a particular challenge for existing robust methods. The fact that AT may sometimes overfit has been previously observed by [177, 182, 217]; in particular, Rice et al. [182] also experiments with a few generalization methods (but *not* dropout) and observes that only early stopping helps overcome overfitting to a certain degree. We expand the scope of these results to show that RST and TRADES also suffer from large generalization gaps, and that dropout can help narrow the gap in these two methods. Furthermore, we demonstrate that dropout often decreases the local Lipschitz constant.

3.6 Related Work

There is a large body of literature on developing adversarial attacks as well as defenses that apply to neural networks [39, 138, 140, 144, 163, 167, 208, 219, 226, 246]. While some of this work has noted that an increase in robustness is sometimes accompanied by a loss in accuracy, the phenomenon remains ill-understood. Raghunathan et al. [177] provides a synthetic problem where adversarial training overfits, which we take a closer look at in Appendix B.4. Raghunathan et al. [178] proposes the robust self training method that aims to improve the robustness and accuracy tradeoff; however, our experiments show that they do not completely close the gap particularly when using only labeled data.

Outside of neural networks, prior works suggest that lack of data may be responsible for low robustness [3, 22, 43, 48, 134, 196, 197, 228, 245]. For example, Schmidt et al. [196] provides an example of a linear classification problem where robustness in ℓ_∞ norm requires more samples than plain accuracy, and Wang et al. [228] shows that nearest neighbors would be more robust with more data. Some prior works also suggest that the robustness accuracy tradeoff may arise due to limitations of existing algorithms. Bubeck et al. [36] provides an example where

finding a robust and accurate classifier is significantly more computationally challenging than finding one that is simply accurate, and Bhattacharjee and Chaudhuri [22] shows that certain non-parametric methods do not lead to robust classifiers in the large sample limit. It is also known that the Bayes optimal classifier is not robust in general [21, 171, 183, 222, 228], and it differs from the maximally astute classifier [22, 239].

Prior work has also shown a connection between adversarial robustness and local or global Lipschitzness. For linear classifiers, it is known that imposing Lipschitzness reduces to bounding the norm of the classifier, which in turn implies large margin [143, 234]. Thus, for linear classification of data that is linearly separable with a large margin, imposing Lipschitzness *helps* generalization.

For neural networks, Anil et al. [7], Huster et al. [100], Qian and Wegman [172] provide methods for imposing global Lipschitzness constraints; however, the state-of-the-art methods for training such networks do not lead to highly expressible functions. For local Lipschitzness, Hein and Andriushchenko [94] first showed a relationship between adversarial robustness of a classifier and local Lipschitzness of its underlying function. Following this, Weng et al. [229] provides an efficient way of calculating a lower bound on the local Lipschitzness coefficient. Many works consider a randomized notion of local smoothness, and they prove that enforcing it can lead to certifiably robust classifiers [52, 132, 170, 190].

3.7 Conclusion

Motivated by understanding when it is possible to achieve both accuracy and robustness, we take a closer look at robustness in image classification and make several observations. We show that many image datasets follow a natural separation property and that this implies the existence of a robust and perfectly accurate classifier that can be obtained by rounding a locally Lipschitz function. Thus in principle robustness and accuracy should both be achievable together

on real world datasets.

We investigate the gap between theory and practice by examining the smoothness and generalization properties of existing training methods. Our results show that generalization may be a particular challenge in robust learning since all robust methods that produce locally smooth classifiers also suffer from fairly large generalization gaps. We then experiment with combining robust classification methods with dropout and see that this leads to a narrowing of the generalization gaps.

Our results suggest that the robustness-accuracy tradeoff in deep learning is not inherent, but it is rather a consequence of current methods for training robust networks. Future progress that ensures both robustness and accuracy may come from redesigning other aspects of the training process, such as customized optimization methods [13, 77, 124, 160, 182, 199, 201, 206] or better network architectures [70, 194, 250] in combination with loss functions that encourage adversarial robustness, generalization, and local Lipschitzness. Some recent evidence for improved network architectures has been obtained by Guo et. al. [87], who search for newer architectures with higher robustness from increased model capacity and feature denoising. A promising direction is to combine such efforts across the deep learning stack to reduce standard and adversarial generalization gaps.

3.8 Broader Impact

In this chapter, we have investigated when it is possible to achieve both high accuracy and adversarial robustness on standard image classification datasets. Our motivation is partially to offer an alternative perspective to previous work that speculates on the inevitability of an accuracy-robustness tradeoff. In practice, if there were indeed a tradeoff, then robust machine learning technology is unlikely to be very useful. The vast majority of instances encountered by practical systems will be natural examples, whereas adversaries are few and far between. A

self-driving car will mostly come across regular street signs and rarely come across adversarial ones. If increased robustness necessitates a loss in performance on natural examples, then the system's designer might be tempted to use a highly accurate classifier that is obtained through regular training and forgo robustness altogether. For adversarially robust machine learning to be widely adopted, accuracy needs to be achieved in conjunction with robustness.

While we have backed up our theoretical results by empirically verifying dataset separation, we are also ready to point out the many limitations of current robustness studies. The focus on curated benchmarks may lead to a false sense of security. Real life scenarios will likely involve much more complicated classification tasks. For example, the identification of criminal activity or the maneuvering of self-driving cars depend on a much broader notion of robustness than has been studied so far. Perturbations in ℓ_p distance cover only a small portion of the space of possible attacks.

Looking toward inherent biases, we observe that test accuracy is typically aggregated over all classes, and hence, it does not account for underrepresentation. For example, if a certain class makes up a negligible fraction of the dataset, then misclassifying these instances may be unnoticeable when we expect a drop in overall test accuracy. A more stringent objective would be to retain accuracy on each separate class, as well as being robust to targeted perturbations that may exploit dataset imbalance.

On a more positive note, we feel confident that developing a theoretically grounded discussion of robustness will encourage machine learning engineers to question the efficacy of various methods. As one of our contributions, we have shown that dataset separation guarantees the existence of an accurate and robust classifier. We believe that future work will develop new methods that achieve robustness by imposing both Lipschitzness and effective generalization. Overall, it is paramount to close the theory-practice gap by working on both sides, and we stand by our suggestion to further investigate the various deep learning components (architecture, loss function, training method, etc) that may compound the perceived gains in robustness and

accuracy.

3.9 Acknowledgements

Kamalika Chaudhuri and Yao-Yuan Yang thank NSF under CIF 1719133, CNS 1804829 and IIS 1617157 for support. Hongyang Zhang was supported in part by the Defense Advanced Research Projects Agency under cooperative agreement HR00112020003. The views expressed in this work do not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. Approved for public release; distribution is unlimited. This work was also supported in part by NSF IIS1763562 and ONR Grant N000141812861.

This chapter is a reformatted version of the material in the manuscript "A Closer Look at Accuracy vs. Robustness" by Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri [240]. The manuscript is published in the *Conference on Neural Information Processing Systems*, 2020. The dissertation author is the co-primary researcher and co-author of the paper.

Chapter 4

Adversarial Robustness for Non-parametric Classifiers

4.1 Introduction

State-of-the-art classifiers have been shown to suffer from substantial drops in accuracy when faced with adversarially modified inputs even if the modifications are imperceptibly slight. Due to the security concerns that this raises, a body of recent research has investigated the construction and prevention of adversarial examples – small perturbations of valid inputs that cause misclassification [38, 219]. Most previous work has looked at parametric methods, i.e., neural networks and linear classifiers [23, 140, 144, 165], and there is a mature understanding of what properties can be exploited to design adversarial attacks and defenses for any parametric model. For example, parametric classifiers are based on continuous functions with gradients, which has been used to design gradient-based attacks [10, 39]. Likewise, parametric models are mostly trained by minimizing a training loss, which has been exploited to build an effective and generic defense – adversarial training, retraining after data augmentation with adversarial examples [40, 144, 211].

An alternative statistical paradigm is that of non-parametric methods, such as nearest neighbor, decision tree, and random forest classifiers, which typically apply to dense data in lower dimensional spaces. These are local predictors, whose output depends on labeled points close to an input. Surprisingly, these methods behave very differently from parametrics when it comes to adversarial examples. In many cases, they have no gradients, and adversarial examples for parametric models fail to transfer [164]. Generic defenses, such as adversarial training, appear to be ineffective as well [63, 161, 228].

While prior work has constructed attacks and defenses for some specific classifiers [47, 63, 108, 209, 228], there appear to be no generic approaches, and no generic principles that can be used to guide the design of attacks and defenses for variety of non-parametric methods.

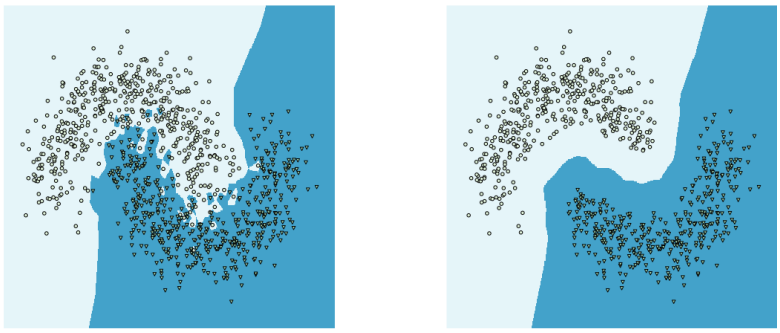


Figure 4.1: Normal vs. Defended 1-Nearest Neighbor.

In this work, we identify two key general principles, and use them to design a generic defense and an attack that apply to a variety of non-parametric methods.

To design defenses, we ask: when do non-parametric methods work well? Figure 4.1 depicts two variants of random forests. In the left figure, we observe that datasets with nearby oppositely-labeled points may lead to classifiers with convoluted decision boundaries. In the right figure, we see that well-separated data lead to classification regions that are more robust to small perturbations. We will use this low-dimensional intuition as a starting point for generic defense

methods.

Figure 4.1 suggests that since these methods make local predictions, they might work well when data from different classes are well-separated in space. We clearly cannot hope for such separation in most real datasets. Therefore, we propose to preprocess the training data by removing a subset so that different classes are well-separated. To ensure classification accuracy, we propose removing the minimal subset of points that ensure this property. We call our method *Adversarial Pruning*, which can be used as a pre-processing step before training any generic non-parametric classifier.

To evaluate our defense, we propose a new attack that is based on our next key observation: many non-parametric methods divide the instance space into convex polyhedra, and predict in a piecewise constant manner in each. For example, for 1-nearest neighbor, these polyhedra are the Voronoi cells. This suggests the following attack: find the closest polyhedron to an input where the classifier predicts a different label and output the closest point in this region. We implement this strategy by solving a collection of convex programs, and in cases where solution is computationally expensive, we provide a heuristic method for finding an approximate solution. We refer to these attacks as the exact and approximate *region-based attack*.

We next provide some theoretical justification for our methods. For our defense, we show that adversarial pruning can be interpreted as a finite-sample version of a robust analogue to the Bayes Optimal classifier. We formally introduce this robust classifier, that we call the r-optimal classifier, and show that it maximizes *astuteness* (accuracy where it is robust with radius r). For our attack, we show that the exact region-based attack is optimal, in the sense that it yields the closest adversarial example to a test input.

We empirically evaluate the adversarial pruning defense using the region based attack and prior attacks. We provide a general and thorough evaluation, for k-nearest neighbors (k-NN), decision trees, and random forests. We see that adversarial pruning consistently improves robustness, outperforming adversarial training on several datasets and is competitive with classifier-specific

defenses. For our attacks, we see that even without any classifier-specific optimization, our new attacks either outperform or are competitive with prior attacks (in terms of perturbation amount). This suggests that both the adversarial pruning defense as well as the region based attack are good generic baselines for evaluating the robustness of non-parametric methods.

4.2 Preliminaries

In this chapter, we follow the multi-class classification and adversarial robustness setup in Section 2.1. We focus on the non-parametric classifiers, which are local classifiers whose output depends on training data close to the test instance. These classifiers are typically used with dense lower-dimensional data, such as those in Figure 4.1. Examples are k-nearest neighbor (k-NN) and tree-based classifiers. The *k-NN classifier* outputs the plurality label among the k training examples closest to $\mathbf{x}$ in an ℓ_p metric. A *tree ensemble* contains T decision trees whose leaves are labeled with vectors in $\mathbb{R}^C$. Each input $\mathbf{x}$ determines T root-to-leaf paths, corresponding to vectors $\mathbf{u}^1, \ldots, \mathbf{u}^T$. The output is the largest coordinate in $\mathbf{u}^1 + \cdots + \mathbf{u}^T$. Random forests are a subclass of tree ensembles.

4.3 Adversarial Pruning Defense

When are non-parametric methods robust? Since these are local classifiers, Figure 4.1 suggests that they may be robust when training data from different classes is well-separated, and may fail when they overlap.

The training data may not be separated, so we will preprocess the data. We remove a subset of the training set, so that the remaining data are well-separated. Then, we train a non-parametric classifier on the rest. A remaining question is which subset of points to remove. For high classification accuracy, we remove the minimum subset whose removal ensures this

property.

This process of removing examples from training set so that certain properties hold is called *pruning*. In this section, we first introduce the method used to prune the dataset. In Section 4.5, we justify our method by interpreting it in light of classical results in statistical learning theory [45, 54, 59].

Formally, given a robustness radius r and training set $\mathcal{D}$, we propose the following generic way to preprocess the training set and improve the robustness of classifiers:

Adversarial Pruning. Given r and a set $\mathcal{D}$, compute a maximum subset $\mathcal{D}^{\mathsf{AP}} \subseteq \mathcal{D}$ such that differently-labeled points have distance at least $2r$. Then, train any non-parametric classifier on $\mathcal{D}^{\mathsf{AP}}$.

After computing $\mathcal{D}^{\mathsf{AP}}$ once for a dataset, then we may train any classifier on the pruned training set. Our main hypothesis is that this will lead to more robust classifiers when using non-parametric methods. We will demonstrate empirically that this works well, and we will argue that this defense method is a finite-sample approximation to the optimal robust classifier.

Observe that while adversarial pruning is similar to the defense in [228], they actually retain additional points with confident labels, which ensures that their method converges to being robust where the *Bayes Optimal* is robust. Their work builds on previous results of [84] and [119] that sharpen the risk analysis of 1-NN by using pruning. As we explain in Section 4.5, our method instead can be interpreted as a finite sample version of a different and more appropriate limit.

One drawback of this approach is that the metric must be fine-grained enough to distinguish between close and far pairs. For most datasets and norms (e.g, Euclidean distance) for which non-parametrics are used, this will be the case. However, for binary features and the ℓ_∞ distance, we have the problem that every pair of different points has distance exactly one, and therefore, the similarity structure is meaningless. To circumvent this, we preprocess the binary feature vectors using standard feature-extraction methods (e.g., PCA), and then operate on the resulting space.

4.3.1 Computing the Robust Dataset

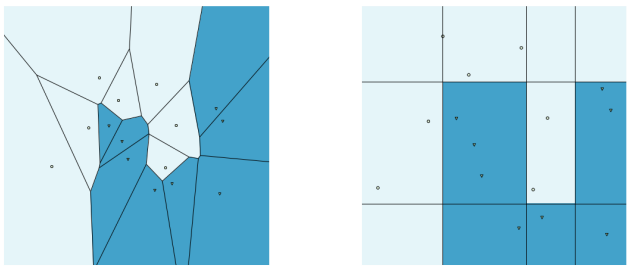
We use known graph algorithms to efficiently compute $\mathcal{D}^{\mathsf{AP}}$. Each training example is a vertex in the graph. Edges connect pairs of differently-labeled examples $\mathbf{x}$ and $\mathbf{x}'$ whenever $\|\mathbf{x}-\mathbf{x}'\| \leq 2r$. We remove as few examples as possible so that no more edges remain. This is equivalent to computing the minimum vertex cover. For binary labels, this graph is bipartite, and a minimum vertex cover can be derived from a maximum matching. The fastest method to solve maximum matching is the Hopcroft-Karp algorithm [96]. For a graph with n vertices and m edges, it takes time $O(m\sqrt{n})$. Fortunately, in practice, the graph of close pairs is quite sparse (for small r and high dimensional feature spaces, with relatively separated classes). For example, if $m = \widetilde{O}(n)$, then computing $\mathcal{D}^{\mathsf{AP}}$ takes time $\widetilde{O}(n^{3/2})$. For large datasets, we note that *linear time* approximation algorithms are known [62].

When there are more than two labels, that is $C \geq 3$, it is NP-Hard to compute the optimal pruned subset, but approximation algorithms are known [84, 119]. The greedy algorithm provably generates a 2-approximation. A suboptimal solution still ensures that different classes are separated, and hence, the robustness of the classifier does not require finding the optimal pruned dataset.

4.4 Region-Based Attack

In this section, we develop a way to evaluate robustness of non-parametric methods. For parametric algorithms, generic gradient-based attacks exist. Our goal is to develop an analogous general attack method, which works well for multiple non-parametrics. Moreover, we aim to develop a white-box attack that will serve as a better baseline than black-box attacks.

The main challenge of finding adversarial examples is that these classifiers have complicated decision regions. The central idea behind our attack is that for many classifiers, such as k-NN or random forests, we can decompose the decision regions into convex sets.



(a) 1-NN (b) Decision tree

Figure 4.2: (s,m)-decompositions of two non-parametrics.

We begin with a brief introduction to non-parametric methods that are local classifiers whose output depends on training data close to the test instance. These methods are typically used with dense lower-dimensional data, such as those in Figure 4.1. Examples are k-nearest neighbor (k-NN) and tree-based classifiers. The *k-NN classifier* outputs the plurality label among the k training examples closest to $\mathbf{x}$ in an ℓ_p metric. A *tree ensemble* contains T decision trees whose leaves are labeled with vectors in $\mathbb{R}^C$. Each input $\mathbf{x}$ determines T root-to-leaf paths, corresponding to vectors $\mathbf{u}^1,\dots,\mathbf{u}^T$. The output is the largest coordinate in $\mathbf{u}^1+\cdots+\mathbf{u}^T$. Random forests are a subclass of tree ensembles.

Definition 9. *An* (s,m)-decomposition *is a partition of* $\mathbb{R}^d$ *into convex polyhedra* $P_1,\dots,P_s$ *such that each* P_i *can be described by up to* m *linear constraints, and* f *is* (s,m)-decomposable *if there is an* (s,m)*-decomposition such that* f *is constant on* P_i *for each* $i\in[s]$.

Figure 4.2 demonstrates the decomposition for two examples. Figure 4.2(a) shows how 1-NN is decomposed. In particular, a Voronoi diagram for n points is an $(n,n-1)$-decomposition ($P_1,\dots,P_n$ are Voronoi cells). If $k\geq 1$, then a k-NN classifier is $\left(\binom{n}{k},k(n-k)\right)$-decomposable; every k points correspond to polyhedra defined by $k(n-k)$ hyperplanes separating the k points from the other $n-k$ points [11].

Tree-based classifiers also fit into our framework, and Figure 4.2(b) shows how a decision

tree is decomposed. Any decision tree of depth D with L leaves is (L, D)-decomposable; each root-to-leaf path corresponds to a polyhedron P_i defined by D hyperplanes. Generally, if f is an ensemble of T trees, each with depth D and L leaves, then f is (L^T, DT)-decomposable (proofs in Appendix C.1). An exponential dependence on T is expected, since the adversarial example problem for tree ensembles is NP-Hard [108].

The existence of (s, m)-decompositions suggests the following attack. Given a classifier f and an input $\mathbf{x}$, suppose we could find the closest polyhedron P_i in the decomposition where f predicts a different label than $f(\mathbf{x})$. Then, the closest point in P_i would be the optimal adversarial example. Our attack implements this strategy by searching over all polyhedra.

4.4.1 Region-Based Attack

Let f be an (s, m)-decomposable classifier with decomposition $P_1, \ldots, P_s$, where $f(\mathbf{z}) = y_i$ when $\mathbf{z} \in P_i$, for labels $y_i \in [C]$. To find an adversarial example for $\mathbf{x}$, consider all polyhedra P_i such that $f(\mathbf{x}) \neq y_i$. Then, output $\widetilde{\mathbf{x}}$ minimizing

$$\min_{i: f(\mathbf{x}) \neq y_i} \; \min_{\mathbf{z} \in P_i} \; \|\mathbf{x} - \mathbf{z}\|. \tag{4.1}$$

Each P_i is described by $\leq m$ linear constraints, and the norm objective is convex [28]. Thus, we can solve each inner minimization problem in Equation (4.1) separately by solving a convex program with $O(m)$ constraints. This results in candidates $\mathbf{z}^i \in P_i$. Taking the outer minimum over i with $f(\mathbf{x}) \neq y_i$ leads to the optimal adversarial example $\widetilde{\mathbf{x}} = \text{argmin}_{\mathbf{z}^i} \; \|\mathbf{x} - \mathbf{z}^i\|$.

Efficiency. The running of the exact attack algorithm depends on two things: (i) the number of regions, which is based on the complexity of the classifier, and (ii) the number of constraints and dimensionality of the polyhedra. Due to advances in linear/quadratic program solvers, finding the adversarial example in a single region is quite efficient, i.e., the inner minimization problem in Equation (4.1) is easy. We find that the number of regions s dominates

the running time, i.e., the outer minimization problem in Equation (4.1) is hard. For k-NN, the number of convex polyhedra scales with $O(n^k)$. When $k = 1$, this is efficiently solvable, because polyhedra have at most n constraints, and the adversarial examples can be found quickly using a linear program for ℓ_∞ perturbations. Unfortunately, for $k > 1$, this attack does not scale well, and we will develop an approximation algorithm for larger values of k.

For a single decision tree, again the exact attack is very efficient, depending only on the number of nodes in the tree. But for larger tree ensembles (e.g., large random forests), the optimal attack is very slow, as expected.

4.4.2 Speeding Up the Search

The exact attack is computationally intensive when s is large; hence, finding optimal solutions is infeasible for random forests (with many trees) or k-NN (when k is large). We next provide a computationally-efficient algorithm, which searches a constant number of regions.

The region-based attack for an (s,m)-decomposable f requires solving up to s convex programs, one for each polyhedron P_i with a different label. If the number of polyhedra is large, then this may be computationally infeasible. Fortunately, Equation (4.1) has an obvious subdivision, based on the outer minimum over convex polyhedra. We use a relaxation that considers only a subset of polyhedra. We observe that each training point corresponds to a polyhedron—the one that f uses to predict the label. When finding adversarial examples for $\mathbf{x}$, the natural choice is to utilize training data close to $\mathbf{x}$.

Approximate Region-Based Attack. Let $\mathcal{D}$ be the training data. To find an adversarial example under ℓ_p for $\mathbf{x}$, we first compute the subset $\mathcal{D}' \subseteq \mathcal{D}$ of s' points closest in ℓ_p distance to $\mathbf{x}$, while having different training labels than $f(\mathbf{x})$. Next, we determine at most s' polyhedra $P_{i_1}, \ldots, P_{i_{s'}}$ containing points in $\mathcal{D}'$ (as the polyhedra partition $\mathbb{R}^d$). We solve the inner optimization problem in Equation (4.1) for each P_{i_j} to find candidates $\mathbf{z}^i$ for $i \in [s']$. Finally, we output $\widetilde{\mathbf{x}} = \text{argmin}_{\mathbf{z}^i} \ \|\mathbf{x} - \mathbf{z}^i\|$, where the minimum is over these s' candidates.

As we only solve $s' \ll s$ convex programs, the running time is greatly reduced compared to the optimal region-based attack. Empirically, this approximation finds adversarial examples with low perturbation.

4.5 Theoretical Justification

We provide some theoretical results to support our methods. To understand the robustness of non-parametric methods, we first derive a theoretically optimal classifier that takes into account robustness as a core objective. Then, we show that adversarial pruning can be interpreted as a finite sample approximation to the optimally robust classifier. Finally, we analyze the exact and approximate region-based attacks.

4.5.1 Adversarial Pruning vs. Optimal

Under certain conditions, many non-parametric methods converge in the infinite sample limit to the *Bayes Optimal classifier*, the most accurate classifier for a data distribution. In this way, non-parametric classifiers may be viewed as finite-sample approximations to the Bayes Optimal. However, the Bayes Optimal may not be robust to adversarial examples.

We next introduce a novel robust analogue to the Bayes Optimal. For a perturbation amount r, we call it the *r-Optimal classifier*. Surprisingly, to the best of our knowledge, such an analogue seems to be new in the context of adversarial examples.

Robust Analogue to Bayes Optimal. In Chapter 2, we introduced the true objective of a robust classifier is the astuteness (Definition 2). We exhibit a classifier, the r-Optimal classifier, that achieves optimal astuteness. It is convenient to rewrite astuteness in terms of certain robust subsets of the input space. Then, we define the r-Optimal classifier using these subsets. Formally,

for a classifier f and label j, let

$$S_j(f,r) := \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = j \text{ and } \rho(f,\mathbf{x}) \geq r\}.$$

We now define the r-Optimal classifier and prove that it maximizes astuteness. This result hinges on the next lemma, which rewrites astuteness in a more convenient form. Let μ be a distribution on labeled examples $\mathcal{X} \times [C]$. The following lemma expresses astuteness under μ using these subsets.

Lemma 4.5.1.

$$\mathsf{ast}_\mu(f,r) = \sum_{j=1}^{C} \int_{\mathbf{x} \in S_j(f,r)} p(y = j \mid \mathbf{x}) d\mu(\mathbf{x}).$$

Proof. Recall the definition of the robust regions of a classifier,

$$S_j(f,r) = \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = j \text{ and } \rho(f,\mathbf{x}) \geq r\}.$$

Starting with the definition of astuteness, we compute the following.

$$\begin{aligned}
&\mathsf{ast}_\mu(f,r) \\
&= \Pr_{(\mathbf{x},y)\sim\mu}[\rho(f,\mathbf{x}) \geq r \text{ and } f(\mathbf{x}) = y] \\
&= \int_{\mathbf{x}} p(y \mid \mathbf{x}) \cdot \mathbf{1}_{\{\rho(f,\mathbf{x})\geq r\}} \cdot \mathbf{1}_{\{f(\mathbf{x})=y\}} \, d\mu(\mathbf{x}) \\
&= \sum_{j=1}^{C} \int_{\mathbf{x}} p(y = j \mid \mathbf{x}) \cdot \mathbf{1}_{\{\rho(f,\mathbf{x})\geq r\}} \cdot \mathbf{1}_{\{f(\mathbf{x})=j\}} \, d\mu(\mathbf{x}) \\
&= \sum_{j=1}^{C} \int_{\mathbf{x} \in S_j(f,r)} p(y = j \mid \mathbf{x}) \, d\mu(\mathbf{x}).
\end{aligned}$$

□

How should we define the classifier that maximizes astuteness? Lemma 4.5.1 implies that, to calculate astuteness, it suffices to consider the robust regions $S_j(f,r)$ for a classifier. As

a consequence, we claim that in order to determine the optimal classifier, it suffices to find the optimal robust regions under μ. We first formalize this intermediate goal using the following maximization problem.

$$\max_{S_1,\ldots,S_C} \sum_{j=1}^{C} \int_{\mathbf{x}\in S_j} p(y=j \mid \mathbf{x})\, d\mu(\mathbf{x}) \tag{4.2}$$
$$\text{s.t. } d(S_j, S_{j'}) \geq 2r \text{ for all } j \neq j',$$

where $d(S_j, S_{j'}) := \min_{u\in S_j, v\in S_{j'}} \|u-v\|$. Notice that for any classifier f, the sets $S_j(f,r)$ for $j \in [C]$ have pairwise distance at least $2r$, implying that they are feasible solutions for Equation (4.2).

Besides being distance $2r$ apart, an optimal solution $S_1^*,\ldots,S_C^*$ to Equation (4.2) maximizes accuracy in the following sense. The integral measures the probability that $(\mathbf{x},y) \sim \mu$ has $y=j$ and $\mathbf{x} \in S_j^*$. In other words, S_j^* has the highest frequency of points with label j under μ, subject to the distance constraint.

The sets S_j^* form the basis for the optimal classifier's decision regions. To ensure the separation, we consider the distance r ball around these sets. Formally, we have the following.

Definition 10. *Fix r and μ. Let $S_1^*,\ldots,S_C^*$ be optimizers of Equation* (4.2)*. The* r-Optimal classifier *f_{ropt} is any classifier such that $f_{\mathsf{ropt}}(\mathbf{x}) = j$ whenever $d(\mathbf{x}, S_j^*) \leq r$.*

We remark that when $r=0$, the 0-Optimal classifier is the standard Bayes Optimal classifier.

Finally, because $S_j(f_{\mathsf{ropt}}, r) = S_j^*$, Lemma 4.5.1 then implies that r-Optimal classifier maximizes astuteness:

Theorem 4.5.2. $f_{\mathsf{ropt}} = \operatorname{argmax}_f \mathsf{ast}_\mu(f,r)$.

Proof. Recall that the r-Optimal classifier f_{ropt} is defined in terms of an optimal solution $S_1^*,\ldots,S_C^*$ to the maximization problem in Equation (4.2). By definition, $f_{\mathsf{ropt}}(\mathbf{x}) = j$ whenever $d(S_j^*, \mathbf{x}) \leq r$. In other words, $S_j^* = S_j(f_{\mathsf{ropt}}, r)$.

We will need the fact that for any classifier f, the sets $S_j(f,r)$ are a feasible solution to the above maximization problem. That is, for $j \neq j'$, the distance between $S_j(f,r)$ and $S_{j'}(f,r)$ is at least $2r$. To see this, consider any two points $u \in S_j(f,r)$ and $v \in S_{j'}(f,r)$. Then, consider the line segment between them $w = \lambda u + (1-\lambda)v$, for $\lambda \in [0,1]$. By definition of the robustness radius (Definition 4), we know that $f(w) = f(u) = j$ whenever $d(w,u) \leq r$. Similarly, $f(w) = f(v) = j'$ whenever $d(w,v) \leq r$. Therefore, we must have that $d(u,v) \geq 2r$. As u and v were an arbitrary pair of points in $S_j(f,r)$ and $S_{j'}(f,r)$, we conclude that these subsets have distance at least $2r$, and this holds for all $j \neq j'$.

Using Lemma 4.5.1, we now compute the following.

$$\begin{aligned}
\mathsf{ast}_\mu(f,r) &= \sum_{j=1}^{C} \int_{\mathbf{x} \in S_j(f,r)} p(y=j \mid \mathbf{x})\, d\mu(\mathbf{x}) \\
&\leq \sum_{j=1}^{C} \int_{\mathbf{x} \in S_j^*} p(y=j \mid \mathbf{x})\, d\mu(\mathbf{x}) \\
&= \sum_{j=1}^{C} \int_{\mathbf{x} \in S_j(f_{\mathsf{ropt}},r)} p(y=j \mid \mathbf{x})\, d\mu(\mathbf{x}) \\
&= \mathsf{ast}_\mu(f_{\mathsf{ropt}},r).
\end{aligned}$$

The inequality uses that the sets $S_j(f,r)$ have pairwise distance at least $2r$, and therefore, they are feasible for the above maximization problem, which has optimal solution $S_j^* = S_j(f_{\mathsf{ropt}},r)$.

□

Finite Sample Approximation. Prior work shows that 1-NN applied to a variant of adversarial pruning leads to provably robust classifiers [228]. The main difference with our work is their method also selects a subset of confident training examples to keep in the pruned subset - which ensures that the classifier converges to being robust in regions where the Bayes Optimal is robust. In contrast, our aim is to develop generic techniques, for multiple classifiers, and we show that our method can be interpreted as a finite sample approximation to the r-Optimal classifier – the optimally astute classifier.

Adversarial pruning works by removing certain training points so that no oppositely labeled pairs of examples remain. We can view this process in the light of the r-optimal classifier as follows. To prune the dataset $\mathcal{D}$, we solve the maximization problem:

$$\max_{S_1,\ldots,S_C \subseteq \mathcal{D}} \sum_{j=1}^{C} \sum_{\mathbf{x}^i \in S_j} \mathbf{1}_{\{y^i = j\}} \tag{4.3}$$

$$\text{s.t. } d(S_j, S_{j'}) \geq 2r \text{ for all } j \neq j'.$$

The solution to Equation (4.3) will be maximum subsets of training data with pairwise distance $2r$. As long as the training set $\mathcal{D}$ is representative of the underlying distribution μ, these subsets will approximate the optimal S_j^* sets. Hence, we posit that a non-parametric method trained on $\mathcal{D}^{\mathsf{AP}}$ should approximate the r-Optimal classifier.

4.5.2 Attack Algorithm Analysis

The run time of the region-based attack depends on the norm. We focus on ℓ_p with $p \in \{1, 2, \infty\}$ as these are the most relevant for adversarial examples. We prove the following theorem in Appendix C.1.

Theorem 4.5.3. *If f is (s,m)-decomposable, then the region-based attack outputs optimal adversarial examples in time $s \cdot \text{poly}(m,d)$, for ℓ_p distance, $p \in \{1,2,\infty\}$.*

As k-NN and tree ensembles are (s,m)-decomposable, the region-based attack produces an optimal adversarial example for these. Note that an optimal attack *certifies* the robustness radius. Indeed, if on input $\mathbf{x}$ the region-based attack outputs $\widetilde{\mathbf{x}}$, then $\|\mathbf{x} - \widetilde{\mathbf{x}}\| = \rho(f, \mathbf{x})$.

We leave it as an interesting open question to develop provably optimal algorithms with better running time. For example, in the case of large tree ensembles, the attack searches over all combinations of one leaf from each tree. This seems wasteful, as many of these polyhedra may be empty (in fact, we find that most potential regions are infeasible for random forests trained on

real datasets).

Approximate Attack Guarantees. We claim that the approximate region-based attack outputs a valid adversarial example when f is (s,m)-decomposable. Each region is defined by m constraints, and f is constant on each region. We search in s' regions, finding the best candidate $\mathbf{z}^i$ from each. Each considered region contains a training example with a different label than $f(\mathbf{x})$. Therefore, the best adversarial example $\widetilde{\mathbf{x}}$ in that region receives a different label $f(\widetilde{\mathbf{x}}) \neq f(\mathbf{x})$. The analysis of the time complexity for finding candidates is $\text{poly}(m,d)$ for each region P_i. Compared to the exact attack (Theorem 4.5.3) we only consider s' regions, so the total time is only $s' \cdot \text{poly}(m,d)$. We find in practice that $s' = 50$ regions suffices for a good attack, and the time only scales with m and d.

4.6 Experiments

We investigate the effectiveness of our methods by evaluating multiple classifiers on nine datasets. We address the following questions:

1. Does adversarial pruning increase robustness across multiple non-parametric classifiers?

2. How well does the region-based attack perform compared with prior work?

4.6.1 Experimental Setup

Classifiers and Datasets. We evaluate three non-parametric classifiers: k-nearest neighbor (k-NN), decision tree (DT) and random forest (RF) [29, 30, 54]. We use nine standard binary classification datasets. All features are scaled to be in [0,1]. We evaluate in ℓ_∞ to be consistent with prior work. We reduce the feature dimension of the image datasets (f-mnist and mnist) with PCA to 25 dimensions for two reasons: (i) non-parametrics are normally used for low dimensional spaces, (ii) adversarial pruning requires non-binary features for ℓ_∞. Details are in Appendix C.2 and

Table 4.1: The Empirical Robustness for different attacks on 1-NN and 3-NN **(lower is better; best is in bold)**.

	1-NN					3-NN			
	Direct	BBox	Kernel	RBA Exact	RBA Appr.	Direct	BBox	Kernel	RBA Appr.
austr.	.442	.336	.379	**.151**	**.151**	.719	.391	.464	**.278**
cancer	.223	.364	.358	**.137**	**.137**	.329	.376	.394	**.204**
covtype	.130	.199	.246	**.066**	.067	.200	.259	.280	**.108**
diabetes	.074	.112	.165	**.035**	**.035**	.130	.143	.191	**.078**
f-mnist06	.080	.140	.187	**.029**	.030	.129	.169	.202	**.051**
f-mnist35	.187	.244	.259	**.075**	.077	.234	.238	.266	**.094**
fourclass	.109	.124	.137	**.090**	**.090**	.101	.113	.134	**.096**
halfmoon	.070	.129	.102	**.058**	**.058**	.105	.132	.115	**.096**
mnist17	.161	.251	.262	**.070**	.073	.221	.261	.269	**.097**

Table 4.2: The Empirical Robustness for different attacks on DT and RF **(lower is better; best is in bold)**.

	DT			RF	
	Papernot's	BBox	RBA-Exact	BBox	RBA-Approx
australian	.140	.139	**.070**	**.364**	.446
cancer	.459	.334	**.255**	.451	**.383**
covtype	.289	.117	**.070**	.256	**.219**
diabetes	.237	.133	**.085**	**.181**	.184
f-mnist06	.200	.182	**.114**	.222	**.199**
f-mnist35	.287	.168	**.112**	**.201**	.246
fourclass	.288	.197	**.137**	.159	**.133**
halfmoon	.098	.148	**.085**	.182	**.149**
mnist17	.236	.175	**.117**	**.237**	.244

the code for the experiment is available at `https://github.com/yangarbiter/adversarial-nonparametrics/`.

Performance Measures. Besides measuring accuracy, we evaluate the attacks using *empirical robustness* (Definition 6) following prior work [47, 108]. To fairly compare classifiers having different accuracies, we actually compute $\mathsf{ER}(A, f, S, t)$ over t test inputs. To do so, we draw t random samples S_t from S that are classified correctly by f, and we report the average of $\mathsf{ER}(A, f, \mathbf{x})$ over $\mathbf{x} \in S_t$. We set $t = 100$ to balance efficiency and thoroughness.

Again, for defenses, we use perturbation distance to evaluate robustness. Each defense method D produces a classifier f_D. We evaluate a defense D by assigning it a score, the defscore (Definition 6). Whenever feasible, we use the optimal attack while calculating the defscore.

Attack Algorithms. For 1-NN and DT, we apply the exact region-based attack (RBA-Exact). For 3-NN and RF, the RBA-Exact attack is computationally intensive, and we use the approximate region-based attack (RBA-Approx). For 3-NN, it uses $s' = 50$ polyhedra, and for RF, it uses $s' = 100$ polyhedra. We compare RBA-Exact and RBA-Approx against several baselines. A general attack that applies to all methods is the black-box attack (BBox) [50]; this attack seems to be the state-of-the-art for non-parametrics. For k-NN, we compare against two white-box attacks, the direct attack (Direct) and kernel substitution attack (Kernel) [164]. The direct attack perturbs the test instance towards the center of the k nearest oppositely-labeled training examples. The kernel substitution attack uses a soft nearest neighbor to build a substitution model and applies the projected gradient descent attack [123]. For DT, the RBA-Exact attack is optimal, and so is the attack by Kantchelian et al. [108]; we only report RBA-Exact because these achieve the same results. We also evaluate the heuristic DT attack by Papernot et al. [164]. For RF, both optimal attacks are infeasible, and we only evaluate BBox and RBA-Approx.

Defense Methods. For our defense, we train each classifier on the dataset pre-processed with adversarial pruning (AP); we use ℓ_∞ to determine examples to prune. For the separation r of AP, we found that $r = 0.3$ balances robustness vs. accuracy. We set $r = 0.3$ for all datasets (Appendix C.2.4 has other r settings). A generic baseline is adversarial training (AT), where the training data is augmented with examples generated by the corresponding attack algorithm. AT has been reported to be ineffective for 1-NN and boosted decision tree [47, 228], but we include it for completeness. For AT, we retrain the classifier after attacking each training point once; we augment the training data with adversarial examples that are distance at most 0.3 from the original input. The parameter 0.3 matches the parameter r for AP. For 1-NN, an available baseline defense is proposed by Wang et al. [228], but for general k-NN, we are not aware of other defenses. For

Table 4.3: defscore using different defenses for four different classifiers **(higher is better; best is in bold)**. The defscore for undefended classifiers is 1.00 (greater than 1.00 is more robust). We use RBA-Exact for 1-NN and DT, and RBA-Approx for 3-NN and RF. We use RBA-Approx for AT on large datasets.

	1-NN			3-NN		DT			RF		
	AT	WJC	AP	AT	AP	AT	RS	AP	AT	RS	AP
austr.	0.64	**1.65**	**1.65**	0.68	**1.20**	2.36	**5.86**	2.37	1.07	**1.12**	1.04
cancer	0.82	1.05	**1.41**	1.06	**1.39**	0.85	1.09	**1.19**	0.87	**1.54**	1.26
covtype	0.61	**4.38**	**4.38**	0.88	**3.31**	1.47	2.73	**4.51**	1.02	1.01	**2.13**
diabetes	0.83	**4.69**	**4.69**	0.87	**2.97**	0.93	1.53	**2.22**	1.19	1.25	**2.22**
f-mnist06	0.90	1.93	**2.59**	0.88	**1.75**	1.33	2.33	**2.57**	1.04	1.10	**1.77**
f-mnist35	0.83	1.05	**1.19**	0.83	**1.15**	0.97	**3.03**	2.06	0.99	1.23	**1.41**
fourclass	0.93	**3.09**	**3.09**	0.89	**3.09**	1.06	1.23	**3.04**	1.03	1.92	**3.59**
halfmoon	1.05	2.00	**2.78**	0.93	**1.92**	1.54	1.98	**2.58**	1.04	1.01	**1.82**
mnist-17	0.88	1.06	**1.39**	0.80	**1.13**	1.11	**3.97**	1.32	0.88	0.92	**1.26**

DT and RF, we compare against the best known defense algorithm, Robust Splitting (RS) [47]. We set the RS parameter to 0.3 as well.

4.6.2 Results

We separately evaluate attacks and defenses, in Tables 4.1 to 4.3, respectively. We also provide an accuracy vs. perturbation distance experiment in Figure 4.3.

Effectiveness of Attacks. Tables 4.1 and 4.2 exhibit empirical robustness across four undefended classifiers and nine datasets. Recall that a smaller empirical robustness implies a more effective attack. For 1-NN, we see that RBA-Exact works as expected, achieving the smallest empirical robustness. For 3-NN, our RBA-Approx attack is more effective than prior attacks, with a much lower empirical robustness. This indicates that RBA-Approx can be a strong attack for $k > 1$, where previously no consistently effective baseline is known. For DT, RBA-Exact again has the best performance. The improvement in many cases shows that the optimal attack for 1-NN and DT can be significantly better than heuristics, which will lead to a more informative defense evaluation. For RF, RBA-Approx wins on five of the nine datasets, and BBox wins on

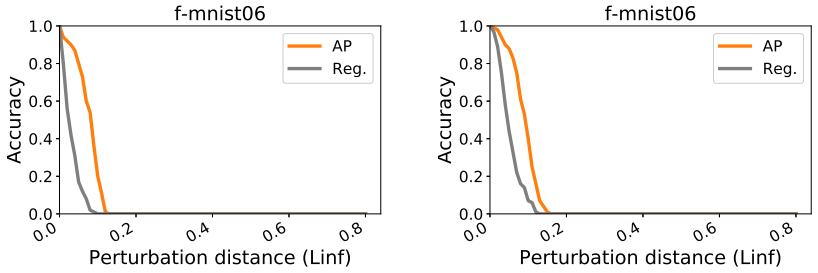
four. Overall, our RBA-Approx attack is competitive with the state-of-the-art attack for RF, and better for 3-NN.

Effectiveness of Defenses. Table 4.3 shows defscore across four classifiers and several defense methods. For each dataset, the AP defense trains all four classifiers on the same pruned version of the dataset. For all classifiers, we see that AP results in a greater than one defscore, indicating that classifiers trained with AP are more robust. In contrast, AT usually achieves defscore less than one, worse than the undefended classifier; this corroborates previous results [228]. For 1-NN, observe that AP is slightly better than the defense of Wang et al. [228]. We believe that this is because their method converges to Bayes Optimal, while AP approximates the r-Optimal classifier. For the DT and RF experiments, we see that RS and AP perform competitively, each winning out on some datasets. Overall, AP performs slightly better than RS. We remark that we have evaluated 1-NN and DT against the optimal attack. This provides concrete evidence that AP leads to a more robust classifier.

4.6.3 Discussion

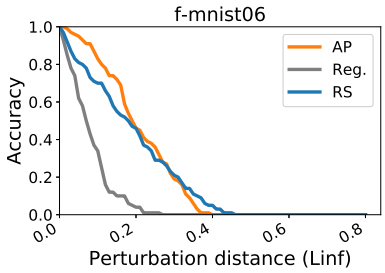
From the results, we see that our generic attack and defense either outperform or perform competitively with prior work on many datasets. We note that there can be a big difference in the perturbation distance depending on the attack algorithms. We also see that our adversarial pruning achieves more robustness compared both to undefended variants and to the classifiers trained using adversarial training. Surprisingly, the pruned subset is computed ahead of time, yet it improves the robustness of many different classifiers.

The main conclusion from the experiments is that our work provides a new and suitable baseline for many methods. This is analogous to how AT and PGD are generic baselines for parametrics. In particular, if a new non-parametric algorithm is developed, then AP and RBA may be used to evaluate robustness. Our work also opens to the door to combine AP with classifier-specific defenses, e.g. robust boosting [47]. We note that our methods can sometimes

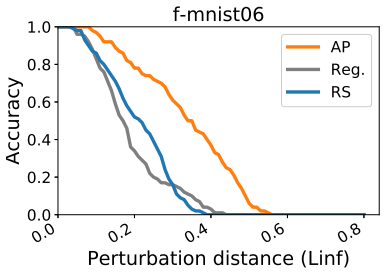


(a) 1-NN

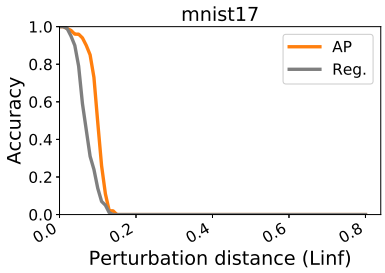
(b) 3-NN



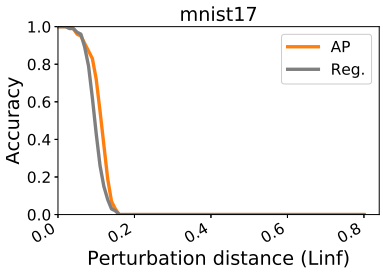
(c) Decision tree



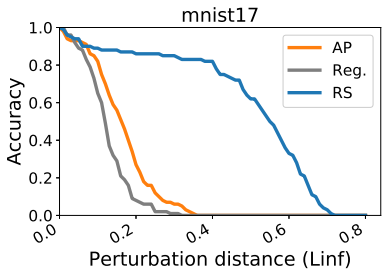
(d) Random forest



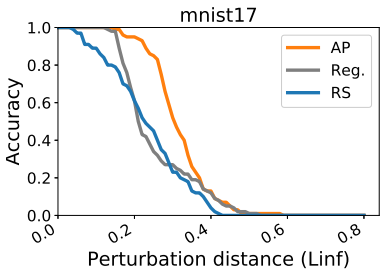
(e) 1-NN



(f) 3-NN



(g) Decision tree



(h) Random forest

Figure 4.3: Accuracy (y-axis) vs. perturbation distance (x-axis) for four classifiers on Fashion MNIST classes 0 vs. 6 (top row, subfigures (a) - (d)) and MNIST classes 1 vs. 7 (bottom row, subfigures (e)-(h)). We used the ℓ_∞ distance after applying PCA to 25 dimensions **(larger accuracy is better)**. Other datasets appear in Appendix C.2.4. In the legend, Reg. = regular (undefended) classifier, AP = adversarial pruning, and RS = robust splitting.

be slow, but we expect that classier-specific optimizations and techniques will readily improve the running time.

4.7 Related Work

The bulk of previous research on robust classifiers has focused on parametric models, with many generic attacks [39, 138, 165, 167, 219], as well as several defenses [93, 109, 144, 162, 176, 208, 244]. In contrast, adversarial examples for non-parametric classifiers have been studied in a more case-by-case basis.

For tree ensembles, Kantchelian et al. [108] formulate an optimal attack as a Mixed Integer Linear Program (superseding an earlier attack [164]) and prove NP-Hardness for many trees. Chen et al. [47] increase the robustness of *boosted* ensembles by introducing a more robust splitting criteria during training. Concurrent work also studies the robustness of decision stumps (i.e., random forests with depth-one trees), and we leave it as future work to compare our methods to theirs [6].

For k-NN, prior work on adversarial examples only considers suboptimal attacks, such the direct attack and variants thereof [5, 209, 228]. Concurrent work [113] on Voronoi-based adversarial training for neural networks also introduces the optimal attack for 1-NN (i.e., Region-Based attack restricted to 1-NN). In terms of defenses, Wang et al. [228] increase 1-NN robustness by strategically removing training points. Besides only testing 1-NN against suboptimal attacks, they do not consider other non-parametrics; additionally, their defense is shown to be robust in the large sample limit only where the Bayes Optimal is robust. Our methods are thus more general, and our defense can be interpreted as a finite sample approximation to the r-Optimal classifier.

Outside the realm of adversarial examples, pruning has been used to improve the accuracy and generalization (but not robustness) of 1-NN [80, 85, 90, 120]. It would be interesting to revisit these works in the context of adversarial robustness, and in particular, in terms of the

r-Optimal classifier.

Related attacks and defenses have been developed for ReLU networks [55, 105, 221, 233]. These results do not directly pertain to non-parametrics, as ReLUs are fundamentally different. The geometric attacks and defenses are similar in spirit to ours. Optimizations based on the dual formulation may improve the efficiency of our methods [221, 233]. It would be interesting to explore the relationship between our defense method (adversarial pruning) and the ReLU defense methods and robustness certificates. For example, do robust ReLU networks approximate or converge to the r-Optimal classifier?

4.8 Conclusion

We consider adversarial examples for non-parametric methods, with a focus on *generic* attacks and defenses. We provide a new attack, the region-based attack, which often outperforms previous attacks. We also provide a new method of defense, adversarial pruning, which should serve as a strong baseline for evaluating the robustness of many classifiers. On the theory side, we prove that the region-based attack outputs the optimal adversarial example. We also introduce and analyze a novel robust analogue to the Bayes Optimal. We prove that the r-Optimal classifier maximizes astuteness. On the experimental side, we demonstrate that our methods are better than or competitive with prior work, while being considerably more general.

4.9 Acknowledgements

We thank Somesh Jha, Ruslan Salakhutdinov, and Michal Moshkovitz for helpful discussions. Part of this research is supported by ONR under N00014-16-1-261, UC Lab Fees under LFR 18-548554 and NSF under 1804829 and 1617157.

This chapter is a reformatted version of the material in the manuscript “Robustness

for Non-Parametric Classification: A Generic Attack and Defense" by Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri [239]. The manuscript is published in the *International Conference on Artificial Intelligence and Statistics*, 2020. The dissertation author is the co-primary researcher and co-author of the paper.

Chapter 5

Adversarial Robustness and its Connection to Interpretability on Decision Trees

5.1 Introduction

Deploying machine learning (ML) models in high-stakes fields like healthcare, transportation, and law, requires the ML models to be trustworthy. Essential ingredients of trustworthy models are interpretability and robustness: if we do not understand the reasons for the model's prediction, we cannot trust the model; if small changes in the input modifies the model's prediction, we cannot trust the model. Previous works hypothesized that there is a strong connection between robustness and interpretability. They empirically observed that robust models lead to better explanations [47, 184]. In this work, we take a rigorous approach towards understanding the connection between robustness and interpretability.

We focus on binary predictions, where each example has d features and the label of each example is in $\{-1,+1\}$, so an ML model is a hypothesis $f : \mathbb{R}^d \to \{-1,1\}$. We want our model to be (i) robust to adversarial ℓ_∞ perturbations, i.e., for a small distortion, $\|\delta\|_\infty$, the model's response is similar, $f(x) = f(x+\delta)$, for most examples x, (ii) interpretable, i.e., the model itself

is simple and so self-explanatory, and (iii) have high-accuracy. A common type of interpretable models are decision trees [149], which we call *tree-based explanation* and focus on in this chapter.

Prior literature [239] showed that data *separation* is a sufficient condition for a robust and accurate classifier. A dataset is r-separated if the distance between the two closest examples with different labels is at least $2r$. Intuitively, if r is large, then the data is well-separated. A separated data guarantees that points with opposite labels are far from each other, which is essential to construct a robust model.

In this chapter, we examine whether separation implies tree-based explanation. We first show that for a decision tree to have accuracy strictly above $1/2$ (i.e., better than random), the data must be bounded. Henceforth, we assume that the data is in $[-1,1]^d$. We start with a trivial algorithm that constructs a tree-based explanation with complexity (i.e., tree size) $2^{O(d/r)}$. For constant r, we show a matching lower bound of $2^{\Omega(d)}$. Thus, we have a matching lower and upper bound on the explanation size of $2^{\Theta(d)}$. Thus, separation implies robustness and interpretability. Unfortunately, for a large number of features, d, the explanation size is too high to be useful in practice.

In this chapter, we show that designing a simpler explanation is possible with a stronger separability assumption — linear separability with a γ-margin. This assumption was recently used to gain a better understanding of neural networks [154, 204, 212]. More formally, this assumption means that there is a vector $\mathbf{w}$ with $\|\mathbf{w}\| = 1$ such that $y\mathbf{w}\cdot\mathbf{x} \geq \gamma$ for each labeled example $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, 1\}$ in the data [202].

One can hope that standard methods for learning linear models will suffice, but this may not be the case. Standard linear models such as ℓ_2-regularized logistic regressions or support vector machines [202] may produce models that use too many features (in other words, the weights are not sparse), and this can make the model not interpretable. Many other approaches [20, 186] try to solve this issue by enforcing sparsity on the weight vector. However, these models may not be adversarially robust. In this chapter, our goal is to find a model that is interpretable, robust,

and a high-accuracy.

Utilizing ideas from [203], we show that under the linearity assumption, there is always at least one feature that provides non-trivial information for the prediction. To formalize this, we use the known notion of *weak learners* [110], which guarantees the existence of hypothesis with accuracy bounded below by more than $1/2$.

The weak-learnability theorem, together with Kearns and Mansour [111], implies that a popular CART-type algorithm [31] provides a decision tree with size $1/\varepsilon^{O(1/\gamma^2)}$ and accuracy $1-\varepsilon$. Therefore, under the linearity assumption, we can design a tree with complexity independent of the number of features. Thus, even if the number of features, d, is large, the interpretation complexity is not affected. This achieves our first goal of constructing an interpretable model with provable guarantees.

Recently, several research papers give a theoretical justification for CART's empirical success [24, 25, 34, 35, 71]. Those papers assume that the underlying distribution is uniform or features chosen independently. For many cases, this assumption does not hold. For example, in medical data, there is a strong correlation between age and different diseases. On the other hand, we give a theoretical justification for CART without resorting to the feature-independence assumption. We use, instead, the linear separability assumption. We believe that this method will allow, in the future, proofs with less restrictive assumptions.

So far, we have shown how to construct an interpretable model, but we want a model that is not just interpretable but also robust. Decision trees are not robust by-default [47]. For example, a slight change in the feature at the root of the decision tree leads to an entirely different model (and thus to entirely different predictions): the model defined by the left subtree and the model defined by the right subtree. We are left with the question, are we able to constrct a tree that is both robust and interpretable. To design such model, we focus on a specific kind of decision tree — risk score [223]. A risk score is composed of several conditions (e.g., $age \geq 75$) and each matched with a weight, i.e., a small integer. A score $s(\mathbf{x})$ of an example $\mathbf{x}$ is the weighted sum of

all the satisfied conditions. The label is then a function of the score $s(\mathbf{x})$. A risk score is a specific case of decision trees, wherein at each level in the tree, the same feature is queried. The number of parameters required to represent a risk score is much smaller than their corresponding decision trees, hence they might be considered more interpretable than decision trees [223].

We design a new learning algorithm, BBM-RS, for learning risk scores that rely on the Boost-by-Majority (BBM) algorithm [74] and our weak learner theorem. It yields a risk score of size $O(\gamma^{-2}\log(1/\varepsilon))$ and accuracy $1-\varepsilon$. Thus, we found an algorithm that creates a risk score with provable guarantees on size and accuracy. As a side effect, note that BBM allows to control the interpretation complexity easily. Importantly, we show that risk scores are also guaranteed to be robust to ℓ_∞ perturbations, by deliberately adding a small noise to dataset (but not too much noise to make sure that the noisy dataset is still linearly separable). Therefore, we design a model that is guaranteed to have high accuracy and be both interpretable and robust, achieving our final goal.

Finally, in Section 5.6, we test the validity of the separability assumption and the quality of the new algorithm on real-world datasets that were used previously in tree-based explanation research. On most of the datasets, less than 12% points were removed to achieve an r-separation with $r \geq 0.05$. For comparison, for binary feature-values $\{-1,1\}$, and ℓ_∞ distance, the best value for r is $r = 1$. The added percentage of points required to be removed for the dataset to be linearly separable is less than 7% on average. Thus, we observe that real datasets are close to being separable and even linearly separable. Then, we explored the quality of our new algorithm, BBM-RS. Even though it has provable guarantees only if the data is linearly separable, we run it on real datasets that do not satisfy this property. We compare BBM-RS to different algorithms learning: decision trees [175], small risk scores [223], and robust decision trees [47]. All algorithms try to maximize accuracy, but different algorithms try to, additionally, minimize interpretation complexity or maximize robustness. None of the algorithms aimed to optimize both interpretability and robustness. We compared the (i) interpretation complexity, (ii) robustness,

and (iii) accuracy of all four algorithms. We find that our algorithm provides a model with better interpretation complexity and robustness while having comparable accuracy.

To summarize, our main contributions are:

Interpretability under separability: optimal bounds. We show lower and upper bounds on decision tree size for r-separable data with $r = \Theta(1)$, of $2^{\Theta(d)}$. Namely, our upper bound proves that for any separable data, there is a tree of size $2^{O(d)}$, and the lower bound proves that separability cannot guarantee an explanation smaller than $2^{\Omega(d)}$.

Algorithm with provable guarantees on interpretability and robustness. Designing algorithms that have provable guarantees both on interpretability, robustness, and accuracy in the context of decision trees is highly sought-after, yet there was no such algorithm before our work. We design the first learning algorithm that has provable guarantees both on interpretability, robustness, and accuracy of the returned model, under the assumption that the data is linearly separable with a margin.

While the CART algorithm is empirically highly effective, its theoretical analysis has been elusive for a long time. As a side effect, we provide an analysis of CART under the assumption of linear separability. To the best of our knowledge, this is the first proof with a distributional assumption that does not include feature independence.

Experiments. We verify the validity of our assumptions empirically and show that for real datasets, if a small percentage of points is removed then we get a linear separable dataset. We also compare our new algorithm to other algorithms that return interpretable models [47, 175, 223] and show that if the goal is to design a model that is both interpretable and robust, then our method is preferable.

5.2 Related Work

Post-hoc explanations. There are two main types of explanations: post hoc explanations

[179] and intrinsic explanations [186]. Algorithms for post hoc explanation take as an input a black-box model and return some form of explanation. Intrinsic explanations are simple models, so the models are self-explanatory. The main advantage of algorithms for post hoc explanations [26, 58, 117, 133, 141, 142, 180, 181] is that they can be used on any model. However, they host a variety of problems: they introduce a new source of error stemming from the explanation method [186]; they can be fooled [128, 210]; some explanations methods are not robust to common pre-processing steps [114], and can be independent both of the model and the data generating process [2]. Because of the critics against post hoc explanations, in this chapter, we focus on intrinsic explanations.

Explainability and robustness. Prior studies research the intersection of explanation and robustness of black-box models [129], decision trees [6, 47], and deep neural networks [83, 144, 184, 219]. Unfortunately, the quality of these methods are only verified empirically. On the theoretical side, most works analyzed explainability and robustness separately. Explainability was researched for supervised learning [78, 79, 97, 146] and unsupervised learning [76, 126, 151]. For robustness, Cohen et al. [52] showed that the technique of randomized smoothing has robustness guarantees. Ignatiev et al. [102] connected adversarial examples and a different type of explainability from the point of view of formal logic.

Risk scores. Ustun and Rudin [223] designed a new algorithm for learning risk scores by solving an appropriate optimization problem. They focused on constructing an interpretable model with high accuracy and did not consider robustness, as we do in this work.

5.3 Preliminaries

We investigate models that are (i) with high-accuracy, (ii) robust, and (iii) interpretable, as formalized next.

High accuracy. We consider the task of binary classification over a domain $\mathcal{X} \subseteq \mathbb{R}^d$. Let μ be an underlying probability distribution[1] over labeled examples $\mathcal{X} \times \{-1,+1\}$. The input to a learning algorithm $\mathcal{A}$ consists of a labeled sample $S \sim \mu^m$, and its output is a hypothesis $h : \mathcal{X} \to \{-1,+1\}$. The accuracy of h is equal to $\Pr_{(\mathbf{x},y)\sim\mu}(h(\mathbf{x}) = y)$. The sample complexity of $\mathcal{A}$ under the distribution μ, denoted $m(\varepsilon,\delta) : (0,1)^2 \to \mathbb{N}$, is a function mapping desired accuracy ε and confidence δ to the minimal positive integer $m(\varepsilon,\delta)$ such that for any $m \geq m(\varepsilon,\delta)$, with probability at least $1-\delta$ over the drawn of an i.i.d. sample $S \sim \mu^m$, the output $A(S)$ has accuracy of at least $1-\varepsilon$.

Robustness. In this chapter, We follow the adversarial robustness setup in Chapter 2 and focus on the ℓ_∞ distance as the distance metric.

Interpretability. We focus on intrinsic explanations, also called interpretable models [186], where the model itself is the explanation. There are several types of interpretable models, e.g., logistic regression, decision rules, and anchors [149]. One of the most fundamental interpretable models, which we focus on in this chapter, is *decision trees* [175]. In a decision tree, each leaf corresponds to a label, and each inner node corresponds to a threshold and a feature. The label of an example is the leaf's label of the corresponding path.

In this chapter, we focus on a specific type of decision trees, *risk score* [224]; see Table 5.1. Risk score is defined by a series of m conditions and a weight for each condition. Each condition compares one feature to a threshold, and the weights should be small integers. A score, $s(\mathbf{x})$, of an example $\mathbf{x}$ is the number of satisfied conditions out of the m conditions, each multiplied by the corresponding weight. The prediction of the risk model f is the sign of the score, $f(\mathbf{x}) = sign(s(\mathbf{x}))$. A risk score can be viewed as a decision tree where at each level there is the same feature-threshold pair. Since the risk-score model has fewer parameters than the corresponding decision tree, it may be considered more interpretable.

[1] In the paper, we will assume that μ has additional properties, like separation or linear separation.

Table 5.1: Two risk score models: LCPA [224] and our new BBM-RS algorithm on the bank dataset [150]. Each satisfied condition is multiplied by its weight and summed. Bias term is always satisfied. If the total score > 0, the risk model predicts "1" (i.e., the client will open a bank account after a marketing call). All features are binary (either 0 or 1).

feature	weights		
	LCPA	BBM-RS	
Bias term	-6	-7	+ ...
Age ≥ 75	-	2	+ ...
Called in Q1	1	2	+ ...
Called in Q2	-1	-	+ ...
Called before	1	4	+ ...
Previous call was Successful	1	2	+ ...
Employment variation rate < -1	5	4	+ ...
Consumer price index ≥ 93.5	1	-	+ ...
3 month euribor rate ≥ 200	-2	-	+ ...
3 month euribor rate ≥ 400	5	-	+ ...
3 month euribor rate ≥ 500	2	-	+ ...
		total score =	

5.4 Separation and Interpretability

We want to understand whether separation implies the existence of a small tree-based explanation. Our first observation is that the data has to be bounded for a tree-based explanation to exist. If the data is unbounded, then to achieve a training error slightly better than random, the tree size must depend on the size of the training data (see Section 5.4.1 and Theorem 5.4.1).

In Section 5.4.2, we investigate lower and upper bounds for decision tree's size, assuming separation. Specifically, in Theorem 5.4.2, we show that if the data is bounded, in $[-1,1]^d$, then r-separability implies a tree based-explanation with tree depth $O(d/r)$. Importantly, the depth of the tree is independent of the training size, so a tree-based explanation exists. Nevertheless, even for a constant r, the size of the tree is exponential in d. In Theorem 5.4.3, we show that this bound is tight as there is a 1-separable dataset that requires an exponential size to achieve accuracy even negligibly better than random. To conclude, if all we know is that the data is r-separability for constant r, the interpretation complexity is $2^{\Theta(d)}$. Unfortunately, this explanation has size

exponential in d. In Section 5.5, we improve the interpretation complexity by assuming a stronger separability assumption. We will assume linear separability with a margin. All proofs are in Appendix D.1.1.

5.4.1 Bounded

In Theorem 5.4.1, we show that the data has to be bounded for a small decision tree to exist. In fact, boundedness is necessary, even if the data is constrained to be linearly separable. For any tree size s and a given accuracy, we can construct a linearly-separable dataset such that any tree of size s cannot have the desired accuracy.

Theorem 5.4.1. *For any tree size s and $\gamma > 0$, there is a dataset in $\mathbb{R}^2$ that is linearly separable, and any decision tree with size s has accuracy less than $\frac{1}{2} + \gamma$.*

5.4.2 Upper and lower bounds

Assuming the data in $[-1,1]^d$ is r-separated, Theorem 5.4.2 tells us that one can construct a decision tree with depth $6d/r$ and training error 0 (and from standard VC-arguments also accuracy $1-\varepsilon$, with enough examples). Importantly, the depth of the tree is independent of the training size n. Nevertheless, it means the size of the trees is exponential in d. The idea of the proof is to fine-grain the data to bins of size about r, in each coordinate. From this construction, it is clear that the returned model is robust at any training data.

Theorem 5.4.2. *For any labeled data in $[-1,1]^d \times \{-1,1\}$ that is r-separated, there is a decision tree of depth at most $\frac{6d}{r}$ which has a training error 0.*

Theorem 5.4.3 proves a matching lower bound by constructing a dataset such that any tree that achieves error better than random, the tree size must be exponential in d. The dataset proving this lower bound is parity. More specifically, it contains the points $\{-1,+1\}^d$ and the label of each point $\mathbf{x}$ is the xor of all of its coordinates.

Theorem 5.4.3. *There is a labeled dataset in* $[-1,1]^d$ *which is* 1*-separated and has the following property. For any* $\gamma > 0$ *and any decision tree* T *that achieves accuracy* $0.5+\gamma$*, the size of* T *is at least* $\gamma 2^d$.

5.5 Linear Separability

In the previous section, we showed that $\Theta(1)$-separability implies a decision tree with size exponential in d, and we showed a matching lower bound. This section explores a stronger assumption than separability that will guarantee a smaller tree, i.e., a simpler explanation. This assumption is that the data is linearly separable with a margin. More formally, data is γ-linearly separable if there is $\mathbf{w} \in \mathbb{R}^d$, $\|\mathbf{w}\|_1 = 1$, such that for each positive example $\mathbf{x}$ it holds that $\mathbf{w} \cdot \mathbf{x} \geq \gamma$ and for each negative example $\mathbf{x}$ it holds that $\mathbf{w} \cdot \mathbf{x} \leq -\gamma$. Note that without loss of generality $w_i \geq 0$ (if the inequality does not hold, multiply the i-th feature in each example by -1). Thus, we can interpret $\mathbf{w}$ as a distribution over all the features. Linear separability might seem at first like a strong assumption, but besides being a widespread assumption [154, 204, 212], in Section 5.6 we show that this assumption is reasonable for real datasets.

As a first attempt, one might hope that $\mathbf{w}$ is a good explanation, but this explanation might use all the features, and the corresponding tree-based explanation might be of exponential size. As a second attempt, one might take the highest $\mathbf{w}_i$'s, since one might interpret the highest $\mathbf{w}_i$ as the most important feature. However, this can be misleading. For example, if all data has the same value at the i-th feature, this feature is meaningless. In this section, we explore a different approach for constructing an interpretable model.

One of our key ideas is to use boosting method [195] to construct a model which is both interpretable, robust, and accurate. This will allow us to gradually add features to the model until we achieve a high-accuracy model. To implement this idea, we show that one feature can provide a nontrivial prediction. In particular, in Section 5.5.1, we show that the hypotheses class,

$\mathcal{H}_t = \{h_{i,\theta}\}$, is a weak learner, where

$$h_{i,\theta}(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{x}_i \geq \theta \\ -1 & \text{o.w.} \end{cases}$$

This class is similar to the known decision stumps class, but it does not contain hypotheses of the form "if $\mathbf{x}_i \leq \theta$ then $+1$ else -1". The reason will become apparent in Section 5.5.3, but for now, we will hint that it helps achieve robustness.

In Section 5.5.2, we observe that weak learnability immediately implies that the known CART algorithm constructs a tree of size independent of d [111]. Unfortunately, decision trees are not necessarily robust. To overcome this difficulty, we focus on one type of decision trees, risk scores, which are interpretable models on their own. In Section 5.5.3 we show how to use the boosting by majority (BBM) [74] algorithm together with our weak learnability theorem to construct a risk score model. We also show that this model is robust. This concludes our quest of finding a model that is *guaranteed* to be robust, interpretable, and have high-accuracy under the linearity separable assumption. In Section 5.6 we will evaluate the model on several real datasets.

5.5.1 Weak learner

This section shows that under the linearity assumption, we can always find a feature that gives nontrivial information, which is formally defined using the concept of a *weak learner* class. We say that a class $\mathcal{H}$ is a weak learner if for every distribution μ over the examples and a function f that are γ-linearly separable, there is hypothesis $h \in \mathcal{H}$ such that $\Pr_{\mathbf{x}\sim\mu}(h(\mathbf{x}) = f(\mathbf{x}))$ is strictly larger than $1/2$, preferably at least $1/2 + \Omega(\gamma)$. Finding the best hypothesis in $\mathcal{H}_t$ can be done efficiently using dynamic programming [202]. The question is how to prove that there must be a weak learner in $\mathcal{H}_t$.

One might suspect that if the data is linearly separable by the vector w (i.e., for each

labeled example $(\mathbf{x}, y)$ it holds that $y\mathbf{w}\mathbf{x} \geq \gamma$), then h_i which corresponds to the highest $\mathbf{w}_i$ is a weak learner. Conversely, if $\mathbf{w}_i$ is small, then the corresponding hypotheses h_i will have a low accuracy. These claims are not true. To illustrate this, think about the extreme example where $\mathbf{w}_1 = 0$ but $\mathbf{x}_1$ completely predicts the output of any example $\mathbf{x}$. From the viewpoint of $\mathbf{w}$, the first feature is irrelevant, as it does not contribute to the term $\mathbf{w} \cdot \mathbf{x}$, but the first feature is a perfect predictor.

One can prove that there is always a hypothesis in $\mathcal{H}_t$ with accuracy $0.5 + \Omega(\gamma)$ by binarizing the input and applying [203]. More specifically, they formed a different connection between linear separability and weak learning. They view each example in the hypotheses basis, and on this basis, the famous minimax theorem implies that linearity is equivalent to weak learnability. In this chapter, we focus on the case that the data, *in its original form*, is linearly separable. Nonetheless, when the features are binary, the two views, the original and hypotheses bases, coincide.

For completeness, in the Appendix D.1.2, we provide a different proof of Theorem 5.5.1, by viewing $\mathcal{H}_t$ as a graph. Namely, define a bipartite graph where the vertices are the examples and the hypotheses and there is an edge between a hypothesis h and example $\mathbf{x}$ if h correctly predicts $\mathbf{x}$. The edges of the graph are defined so that (i) the degree of the hypotheses vertices corresponds to its accuracy and (ii) the linearity assumption ensures that the degree of the example vertices is high. These two properties of the graph proves the theorem.

Theorem 5.5.1. *Fix $\alpha > 0$. For any data in $[-1,1]^d \times \{-1,1\}$ that is labeled by a γ-linearly separable hypothesis f and for any distribution μ on the examples, there is a hypothesis $h \in \mathcal{H}_t$ such that*

$$\Pr_{\mathbf{x}\sim\mu}(h(\mathbf{x}) = f(\mathbf{x})) \geq \frac{1}{2} + \frac{\gamma}{2} - \alpha$$

.

So far, we have shown the existence of hypothesis in $\mathcal{H}_t$ with accuracy $0.5 + \Omega(\gamma)$.

Standard arguments in learning theory imply that the hypothesis that maximizes the accuracy on a sample also has accuracy $0.5+\Omega(\gamma)$. Specifically, for any sample S, denote by h_S the best hypothesis in $\mathcal{H}_t$ on the sample S. Basic arguments in learning theory shows that for a sample of size $m = O\left(d+\log\frac{1}{\delta}/\gamma^2\right)$, the hypothesis h_S has a good accuracy, as the following theorem proves.

Theorem 5.5.2 (weak-learner)**.** *Fix $\alpha > 0$. For any distribution μ over $[-1,+1]^d \times \{-1,+1\}$ that satisfies linear separability with a γ-margin, and for any $\delta \in (0,1)$ there is $m = O\left(\frac{d+\log\frac{1}{\delta}}{\gamma^2}\right)$, such that with probability at least $1-\delta$ over the sample S of size m, it holds that*

$$\Pr_{(\mathbf{x},y)\sim\mu}(h_S(\mathbf{x}) = y) \geq \frac{1}{2} + \frac{\gamma}{4} - \alpha.$$

5.5.2 Decision Tree Using CART

CART is a popular algorithm for learning decision trees. In [111], it was shown that if the internal nodes define a γ-weak learner and number of samples is some polynomial of $t\log(1/\delta)d$, then a CART-type algorithm returns a tree with size $t = 1/\varepsilon^{O(1/\gamma^2)}$ and accuracy at least $1-\varepsilon$, with probability at least $1-\delta$. Under the linearity assumption, we know that the internal nodes indeed define a γ-weak learner by Theorem 5.5.2. Thus, we get a model with a tree size independent of the training size and the dimension. But the model is not necessarily robust.

The above results can be interpreted as proof of the CART algorithm's success. This proof does not use the strong assumption of feature independence, which is assumed in recent works [24, 25, 34, 35, 71].

Designing robust decision trees is inherently a difficult task. The reason is that, generally, the model defined by the right and left subtrees can be completely different. The feature i in the root determines if the model uses the right or left subtrees. Thus, a small change in the i-th feature completely changes the model. To overcome this difficulty, we focus on a specific type of decision tree, risk scores [224], see Table 5.1 for an example. In the decision tree that corresponds to the risk score, the right and left subtrees are the same. In the next section, we design risk scores

that have guarantees on the robustness and the accuracy.

5.5.3 Risk Score

This section designs an algorithm that returns a risk score model with provable guarantees on its accuracy and robustness, assuming that the data is linearly separable. In the previous section, we used [111] that viewed CART as a boosting method. This section uses a more traditional boosting method — the Boost-by-Majority algorithm (BBM) [74]. This boosting algorithm gets as an input training data and an integer T, and at each step $t \leq T$ it reweigh the examples and apply a γ-weak learner that returns a hypothesis $h_t : \mathbb{R}^d \to \{-1,+1\}$. At the end, after T steps, BBM returns $sign\left(\sum_{t=1}^{T} h_t\right)$. In [74, 195] it was shown that BBM returns hypothesis with accuracy at least $1-\varepsilon$ after at most $T = O(\gamma^{-2}\log(1/\varepsilon))$ rounds.

The translation from BBM, which uses $\mathcal{H}_t$ as a weak learner, to a risk score model, is straightforward. The hypotheses in $\mathcal{H}_t$ exactly correspond to the conditions in the risk score. Each condition has weight of 1. If the number of conditions that hold is at least $T/2$ then our risk model returns $+1$, else it returns -1. Together with Theorem 5.5.1 and [74] we get that BBM returns a risk score with accuracy at least $1-\varepsilon$ and with $T = O(\gamma^{-2}\log(1/\varepsilon))$ conditions.

We remark that other boosting methods, e.g., [75, 106], cannot replace BBM in the suggested scheme since the final combination has to be a simple sum of the weak learners and *not* arbitrary linear combination. The letter corresponds to a risk score where the weights are in $\mathbb{R}$ and not a small integer, as desired.

Our next and final goal is to prove that our risk score model is also robust. For that, we use the concept of *monotonicity*. For $\mathbf{x}, y \in \mathbb{R}^d$, we say that $\mathbf{x} \leq y$ if and only if for all $i \in [d]$ it holds that $\mathbf{x}_i \leq y_i$. A model $f : \mathbb{R}^d \to \{0,1\}$ is monotone if for all $\mathbf{x} \leq y$ it holds that $f(\mathbf{x}) \leq f(y)$. We will show that BBM with weak learners from $\mathcal{H}_t$ yields a monotone model. The reasons are (i) all conditions are of the form "$\mathbf{x}_i \geq \theta$", (ii) all weights are non-negative, except the bias term, and (iii) classification of a risk score is detriment by the score's sign. All proofs appear in

Appendix D.1.3.

Claim 5.5.1. *If every condition in a risk-score model R is of the form "$x_i \geq \theta$" and all weights are positive, except the bias term, then R is a monotone model.*

In Claim 5.5.2 we show that, by carefully adding a small noise to each feature, we can transform any algorithm that returns a monotone model to one that returns a robust model.

Claim 5.5.2. *Assume a learning algorithm A gets as an input a sample from a γ-linearly separable data and returns a* monotone *model with accuracy $1-\varepsilon(\gamma)$. Then, there is an algorithm that returns a model with astuteness (Definition 2) at least $1-\varepsilon\left(\frac{\gamma}{2}\right)$ at radius $\gamma/2$.*

To summarize, in Algorithm 1, we show the pseudocode of our new algorithm, BBM-RS. In the first step we add noise to each example by replacing each example $(\mathbf{x}, y)$ by $(\mathbf{x}-\tau y\mathbf{1}, y)$, where $\tau \in (0,1)$ is a parameter that defines the noise level and $\mathbf{1}$ is the all-one vector. In other words, we add noise $y\tau$ to each feature. In the second step, the algorithm iteratively adds conditions to the risk score. At each iteration, we first find the distribution μ defined by BBM [74]. Then, we find the best hypothesis $h_{i,\theta}$ in $\mathcal{H}_t$, according to μ. We add to the risk score a condition "$\mathbf{x}_i \geq \theta$". Finally, we add a bias term of $-T/2$, to check if at least half of the conditions are satisfied.

5.6 Experiments

In previous sections, we designed new algorithms and gave provable guarantees for separated data. We next investigate these results on real datasets. Concretely, we ask the following questions:

- How separated are real datasets?
- How well does BBM-RS perform compared with other interpretable methods?

Algorithm 1 BBM-RS (BBM-Risk Score)

input: D: linearly separable training data by w; WLOG $\forall i.w_i \geq 0$
        T: bound on interpretation complexity
        τ: noise level
output: risk score
# Add noise:
for $(x,y) \in D$ **do**
    replace (x,y) with $(x - \tau y\mathbf{1}, y)$
end for
for $i = 1 \ldots T$ **do**
    $\mu \leftarrow$ BBM distrbution on D
    $i, \theta \leftarrow \text{argmax}_{i,\theta} \sum_{(x,y)\in D} \mu(x) I_{(x_i - \theta)y > 0}$
    Add condition "$x_i \geq \theta$" to RS
end for
Add a bias term of $-T/2$ to RS
return RS

- How do interpretability, robustness, and accuracy tradeoff with one another in BBM-RS?

Datasets. To maintain compatibility with prior work on interpretable and robust decision trees [136, 224], we use the following pre-processed datasets from their repositories – adult, bank, breastcancer, mammo, mushroom, spambase, careval, ficobin, and campasbin. We also use some datasets from other sources such as LIBSVM [44] datasets and Moro et al. [150]. These include diabetes, heart, ionosphere, and bank2. All features are normalized to $[0,1]$. More details can be found in Appendix D.2. The dataset statistics are shown in Table 5.2. Experiment code is available in a public repository[2].

5.6.1 Separation of Real Datasets

To understand how separated they are, we measure the closeness of each dataset to being r- or linearly separated. The *separateness* of a dataset is one minus the fraction of examples needed to be removed for it to be r- or linearly separated.

[2] https://github.com/yangarbiter/interpretable-robust-trees

Table 5.2: Dataset statistics. Columns "sep." records the separateness of each dataset. Columns "$2r$" and "γ" are calculated after the dataset is separated by removing $1 -$ sep points.

	dataset statistics				r-separation		γ-linear separation	
	# samples	# features	# binary features	portion of positive label	sep.	$2r$	sep.	γ
adult	32561	36	36	0.24	0.88	1.00	0.84	0.001
bank	41188	57	57	0.11	0.97	1.00	0.90	0.33
bank2	41188	63	53	0.11	1.00	0.0004	0.91	0.00002
breastcancer	683	9	0	0.35	1.00	0.11	0.97	0.0003
careval	1728	15	15	0.30	1.00	1.00	0.96	0.003
compasbin	6907	12	12	0.46	0.68	1.00	0.65	0.20
diabetes	768	8	0	0.65	1.00	0.11	0.77	0.0008
ficobin	10459	17	17	0.48	0.79	1.00	0.70	0.33
heart	270	20	13	0.44	1.00	0.13	0.89	0.0003
ionosphere	351	34	1	0.64	1.00	0.80	0.95	0.0007
mammo	961	14	13	0.46	0.83	0.33	0.79	0.14
mushroom	8124	113	113	0.48	1.00	1.00	1.00	0.02
spambase	4601	57	0	0.39	1.00	0.000063	0.94	0.000002

For r-separation, we use the algorithm designed by Yang et al. [239] that calculates the minimum number of examples needed to be removed for a dataset to be r-separated with $r \geq 10^{-5}$. This ensures that after removal, there will be no pair of examples that are very similar but with different labels. Finding the optimal separateness for linear separation is NP-hard [19], thus we run a ℓ_1 regularized linear SVM with regularization terms $C = \{10^{-10}, 10^{-8}, \ldots, 10^{10}\}$ and record the lowest training error as an approximation to one minus the optimal separateness.

The separation results are shown in Table 5.2. Eight datasets are already r-separated (separateness $= 100\%$). In the five datasets with separateness $< 100\%$, there are examples with very similar features but different labels. This occurs mostly in binarized datasets; see Appendix D.3 for an example. Three datasets are almost separated with separateness equal to 97%, 88%, and 83%, and two have separateness 68% and 79%. To summarize, 84% of the datasets are r-separated with $r \geq 10^{-5}$, after removing at most 17% of the points.

Linear separation is a stricter property than r-separation, so the separateness for linear separation is smaller or equal to the separateness for r-separation. Seven datasets have separateness

$\geq 90\%$, three separateness between 79% and 89%, and the remaining three have separateness $< 79\%$. After removing the points, all datasets are γ-linearly separable and nine datasets have $\gamma \geq 0.001$. To summarize, (i) 77% of the datasets are close to being linearly separated (ii) requiring linear-separability reduces the separateness of the r-separated dataset by only an average of 6.77%. From this we conclude that for these datasets at least, the assumption of $r-$ or linear-separability is approximately correct.

Table 5.3: Comparison of BBM-RS with other interpretable models. In bold: the best algorithm for each dataset and criterion. Note that several datasets (adult, bank, careval, compasbin, ficobin, and mushroom) have ER = 0.5 for tree-based models (DT, RobDT, and BBM-RS), because these datasets have all binary features and tree-based models set the threshold in the middle of 0 and 1.

	IC (lower=better)				test accuracy (higher=better)				ER (higher=better)			
	DT	RobDT	LCPA	BBM-RS	DT	RobDT	LCPA	BBM-RS	DT	RobDT	LCPA	BBM-RS
adult	414.20	287.90	14.90	**6.00**	**0.83**	**0.83**	0.82	0.81	**0.50**	**0.50**	0.12	**0.50**
bank	30.70	26.80	8.90	**8.00**	**0.90**	**0.90**	**0.90**	**0.90**	**0.50**	**0.50**	0.20	**0.50**
bank2	30.00	30.70	13.80	**4.50**	**0.91**	0.90	0.90	0.90	0.12	0.18	0.10	**0.50**
breastcancer	15.20	7.40	**6.00**	11.00	0.94	0.94	**0.96**	**0.96**	0.23	**0.29**	0.28	0.27
careval	59.30	28.20	10.10	**8.70**	**0.97**	0.96	0.91	0.77	**0.50**	**0.50**	0.19	**0.50**
compasbin	67.80	33.70	**5.40**	7.60	**0.67**	**0.67**	0.65	0.66	**0.50**	**0.50**	0.15	0.33
diabetes	31.20	27.90	6.00	**2.10**	0.74	0.73	**0.76**	0.65	0.08	0.08	0.09	**0.15**
ficobin	30.60	59.60	**6.40**	11.80	0.71	0.71	0.71	**0.72**	**0.50**	**0.50**	0.22	**0.50**
heart	20.30	13.60	11.90	**9.50**	0.76	0.79	**0.82**	**0.82**	0.23	0.31	0.14	**0.32**
ionosphere	11.30	8.60	17.90	**6.80**	0.89	**0.92**	0.88	0.86	0.15	0.25	0.07	**0.28**
mammo	27.40	12.40	7.20	**1.90**	**0.79**	**0.79**	**0.79**	0.77	0.47	**0.50**	0.21	**0.50**
mushroom	10.80	**9.10**	23.80	9.90	**1.00**	**1.00**	**1.00**	0.97	**0.50**	**0.50**	0.10	**0.50**
spambase	153.90	72.30	29.50	**5.60**	**0.92**	0.87	0.88	0.79	0.00	0.04	0.02	**0.05**

5.6.2 Performance of BBM-RS

Next, we want to understand how our proposed BBM-RS performs on real datasets. We compare the performance of BBM-RS with three different baselines on three evaluation criteria: interpretability, accuracy, and robustness.

Baselines. We compare BBM-RS with three baselines: (i) LCPA [224], an algorithm for learning risk scores, (ii) DT [31], standard algorithm for learning decision trees, and (iii) Robust decision tree (RobDT) [47], an algorithm for learning robust decision trees.

We use a 5-fold cross-validation based on accuracy for hyperparameters selection. For

DT and RobDT, we search through $5, 10, \ldots 30$ for the maximum depth of the tree. For BBM-RS, we search through $5, 10, \ldots 30$ for the maximum number of weak learners (T). The algorithm stops when it reaches T iterations or if no weak learner can produce a weighted accuracy > 0.51. For LCPA, we search through $5, 10, \ldots 30$ for the maximum ℓ_0 norm of the weight vector. We set the robust radius for RobDT and the noise level τ for BBM-RS to 0.05. More details about the setup of the algorithms can be found in Appendix D.2.

Evaluation

We evaluate interpretability, accuracy, and robustness of each baseline. The data is randomly split into training and testing sets by 2:1. The experiment is repeated 10 times with different training and testing splits. The mean and standard error of the evaluation criteria are recorded.

Interpretability. We measure a model's interpretability by evaluating its *Interpretation Complexity (IC)*, which is the number of feature-thresholds pairs in the model (one can think of this as the number of tests the model performs). For decision trees (DT and RobDT), the IC is the number of internal nodes in the tree, and for risk scores (LCPA and BBM-RS), the

Table 5.4: The IC of four different methods across all datasets. Here, we use the depth of the tree as the interpretable complexity measure for DT and RobDT.

	DT	RobDT	LCPA	BBM-RS
adult	10.00	12.50	14.90	**6.00**
bank	**5.00**	6.00	8.90	8.00
bank2	5.00	6.00	13.80	**4.50**
breastcancer	6.00	**5.20**	6.00	11.00
careval	12.30	11.40	10.10	**8.70**
compasbin	7.40	7.90	**5.40**	7.60
diabetes	6.00	7.50	6.00	**2.10**
ficobin	**5.00**	7.00	6.40	11.80
heart	**6.00**	6.10	11.90	9.50
ionosphere	**6.00**	7.90	17.90	6.80
mammo	5.60	6.20	7.20	**1.90**
mushroom	**5.80**	6.00	23.80	9.90
spambase	17.40	17.60	29.50	**5.60**

number of non-zero terms in the weight vector. The lower the IC is, the more interpretable the model is. This is a global measure of the models' complexity as we constructed a model which is self-explainable. One can also measure the local complexity of the model, measured by depth.

Robustness. We measure model's robustness by evaluating its *Empirical robustness (ER)* (Definition 5) [239].

Results

The results are shown in Table 5.3 (only the means are shown, the standard errors can be found in Appendix D.3). We see that BBM-RS performs well in terms of interpretability and robustness. BBM-RS performs the best on nine and eleven out of thirteen datasets in terms of interpretation complexity and robustness, respectively. In terms of accuracy, in nine out of the thirteen datasets, BBM-RS is the best or within 3% to the best. These results show that on most datasets, BBM-RS is better than other algorithms in IC and ER while being comparable in accuracy.

In addition to using the number of feature-thresholds pairs as a (global) measure for IC, we also present in Table 5.4 the results in terms of the local measure for IC, i.e., the depth. This local measure considerably favors decision trees (DT and RobDT), since in the same depth, DT and RobDT can use *exponentially* more feature-threshold pairs than LCPA and BBM-RS, which can be much less interpretable. From the table, we see that even in this case, BBM-RS can still have a comparable results with DT and RobDT. In Appendix D.3.4, the standard error of Table 5.4 is recorded.

5.6.3 Tradeoffs in BBM-RS

The parameter τ gives us the opportunity to explore the tradeoff between interpretability, robustness, and accuracy within BBM-RS. Figure 5.1 shows that for small τ, BBM-RS's IC is high, and its ER is low, and when τ is high, IC is low, and ER is high. This empirical observation

strengthens the claim that interpretability and robustness are correlated. See Appendix D.3 for experiments on other datasets and experiments on the tradeoffs between IC and accuracy.

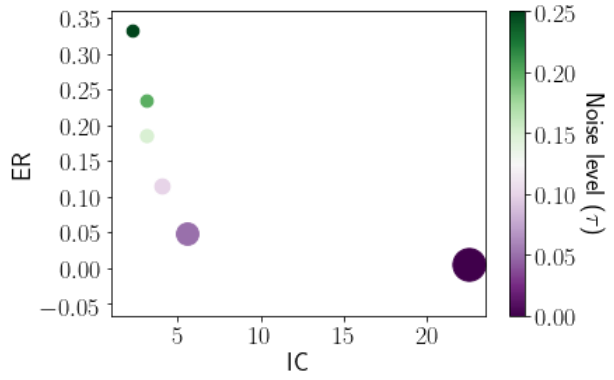


Figure 5.1: Interaction of interpretability, accuracy, and robustness with different noise level τ on the spambase dataset. The size of each ball represents the accuracy. For $\tau = 0$: $IC = 22.5, ER = 0.006$ and for higher noise $\tau = 0.25$: $IC = 2.3, ER = 0.33$

5.7 Conclusion

We found that linear separability is a hidden property of the data that guarantees both interpretability and robustness. We designed an efficient algorithm, BBM-RS, that returns a model, risk-score, which we prove is interpretable, robust, and have high-accuracy. An interesting open question is whether a weaker notion than linear separability can give similar guarantees.

5.8 Acknowledgements

Kamalika Chaudhuri and Yao-Yuan Yang thank NSF under CIF 1719133 and CNS 1804829 for research support.

This chapter is a reformatted version of the material in the manuscript “Connecting Interpretability and Robustness in Decision Trees through Separation” by Michal Moshkovitz, Yao-Yuan Yang, and Kamalika Chaudhuri [152]. The manuscript is published in the *International*

Conference on Machine Learning, 2021. The dissertation author is the co-author of this paper.

Chapter 6

Robustness and Generalization to Nearest Categories

6.1 Introduction

Recently, there has been much interest in various aspects of out-of-distribution (OOD) generalization, such as transfer learning [191], outlier detection [147], and few-shot learning [116]. We want to understand the output of neural networks on OOD inputs, and whether there are patterns in the predicted values. By observing the outputs of a neural network with OOD inputs in Table 6.1, we find that there are indeed some patterns. The question is, what is this pattern? A line of work in the psychology literature posits that humans categorize unseen examples into the most similar category they have seen before [12, 159, 185, 192]. For example, when a child sees an orange for the first time, he may categorize an orange as a type of similar fruit he has seen before, such as a tangerine. Inspired by this unique tendency of humans, we investigate whether neural networks show similar behavior.

We test whether neural networks also tend to predict OOD examples as the nearest category in the training set, and we call this property Nearest Category Generalization (NCG). We

Table 6.1: We remove images of a class from training set of CIFAR10 and CIFAR100, and train a neural network on the modified training set. We then look at the predictions of the neural network on these removed images and record their top two most predicted classes. From the result, we can see that the outputs that these two networks produce follow some patterns. "airplanes" are predicted as a ship or bird possibly because they have similar background of the sky. "aquatic mammals" are predicted as fish possibly because they are both in the water.

	removed class	top most predicted class	second most predicted class
CIFAR10	airplane	ship	bird
CIFAR100	aquatic mammals	fish	small mammals

begin with setting up a framework for testing whether neural networks show signs of NCG. We use images from an unseen class as the OOD examples. We take existing datasets, remove examples of a certain class from the training set, and treat the removed examples as OOD examples. We then train a neural network on the training set and examine its prediction of these OOD examples. If a significant amount of OOD examples are classified as the same category as their nearest training example, then it shows that there are some particular structures in the prediction of the unseen class. We define the *NCG accuracy* as the portion of OOD examples that are predicted as the same label as their nearest training example (while measuring in-distribution accuracy as usual).

Building on this testing framework, we measure the NCG property of neural networks. We consider four datasets and select ten different unseen classes for each dataset. We train a neural network on each of these 40 different combinations of datasets and unseen classes. We find that the NCG accuracies of **all** networks are significantly above the chance levels. This shows concrete evidence that neural networks follow NCG property to predict the unseen class (instead of predicting randomly).

Adversarial robust neural networks are trained to produce smooth predictions when the input is slightly altered. This is another important ability that humans also possess [144, 240, 244]. Does making the network more smooth (or robust) affect their NCG property? We repeat the previous experiment on adversarial robust networks and find that robust networks not only have

an NCG accuracy above chance, but also generally have a higher NCG accuracy than the naturally trained models. This indicates improving adversarial robustness may make the model follow NCG more rigorously, and there are certain connections between adversarial robustness and the NCG property.

Why do robust networks generally have higher NCG accuracies? A plausible explanation for why this may happen is that robust training algorithms like TRADES [244] enforce the network to be locally smooth in a ball of radius r around training data [240]; if the OOD inputs are closer than r from their nearest training example, then they would get classified in the same class. Surprisingly, we find that this is not the case. Balls of radius r around most training examples are so small that they cover almost none of the OOD inputs. Moreover, OOD inputs that are classified with their nearest categories are considerably further than from their closest training examples, which, in turn, continues to have adversarial examples that are closer than the robustness radius r (see Figure 6.1). This suggests that the robust neural networks may be smoother in some directions than others, and perhaps smoother than they were trained to be along the natural image manifold.

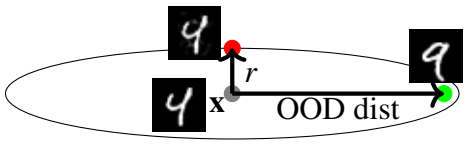
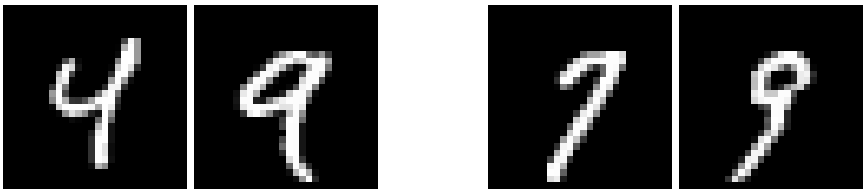


Figure 6.1: Robust networks tend to predict smoothly at a larger distance in some directions, e.g., toward natural OOD examples (green point), but are susceptible to adversarial examples that are closer in the worst-case directions (red point).

A natural question to ask is – does NCG extend to other types of OOD data beyond the unseen classes? To answer the question, we look at the corrupted data including CIFAR10-C, CIFAR100-C, and ImgNet100-C proposed by Hendrycks and Dietterich [95]. We have three observations. First, the NCG accuracies for **all** networks (including natural and robust networks) are above the chance levels, and the NCG accuracies of robust networks are also generally higher than natural networks. This result allows us to extend our previous findings to many kinds of



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

Figure 6.2: OOD examples (b) and (d) are far in pixel space from their nearest training examples (a) and (c). Surprisingly, (b) is predicted as a four and (d) as a seven, indicating the network is smooth in these directions.

OOD data. Second, corrupted examples that are correct in terms of NCG accuracy have a higher chance of being classified correctly. Third, in general, the test and NCG accuracies of a network decreases as the intensity of corruption increases. We find that robust models have a slower rate of decrease comparing with naturally trained models. The second and third observations suggest that different forms of robustness, including adversarial robustness, the robustness to corruptions, and the NCG accuracy, may be inherently interconnected.

In summary, our work uncovers an intriguing out-of-distribution generalization property of neural networks called the nearest category generalization and investigates it in detail. We have identified that the NCG property exists for many neural networks and OOD types. We also show a connection among the NCG property, adversarial robustness, and robustness to corrupted data. We posit that the NCG property is a consequence of the inductive bias produced by neural networks (especially for adversarially robust networks). It is interesting that this inductive bias happens to be similar to some human behaviors and enforcing adversarial robustness, which is another feature that humans possess, can make the NCG property more salient. Many scholars conjecture that the effectiveness of deep learning may be coming from its similar structure to the human brain [91, 127, 198], which allows the neural networks to share some of the inductive biases from the brain. This work can be an additional piece of evidence supporting this theory. In addition, how neural networks generalize so well is still an open question [137]. Our work provides some insights into how networks generalize, and we expect future work to build upon

this knowledge.

6.2 Preliminaries

Nearest Category Generalization. At training time, we are given a set of examples $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$ from one of C categories. At test time, we evaluate on examples drawn from a combination of the training distribution and from a new $(C+1)$-st category. Examples from categories $\{1, 2, \ldots, C\}$ are considered in-distribution and those from class $C+1$ out-of-distribution. For example, we may see the MNIST classes $0-8$ at training time, and all MNIST classes at test time. We call the set of in-distribution test examples the *test set*, and the set of OOD test examples the *OOD set*. In addition to test accuracy, we also look at the NCG accuracy, which is the fraction of inputs from the $(C+1)$-st category that is assigned the same label as its nearest neighbor in the training data. Throughout, we use the shorthand `dataset`-wo# to mean that this class number is the unseen class (category). For example, we let MNIST-wo0 and MNIST-wo9 are MNIST with unseen digits 0 and 9, CIFAR10-wo0 is CIFAR10 with unseen *airplane* and CIFAR100-wo0 is CIFAR100 with unseen *aquatic mammals*. We sometimes shorten this as M-0, C10-0, C100-0, etc.

Distance metric. We need to specify a distance metric for the nearest neighbor. We use ℓ_2 distance in the pixel space, which is a commonly used distance metric; however, it may not provide much semantic information, which is important in some cases. Therefore, in the experiment, we also consider ℓ_2 distance in the feature space. In the pixel space, we use the original image as the input to the neural network. In the feature space, we first train a neural network on the training set (without the unseen class), and then we use this network to extract the features of each image in the training, testing, and OOD set (forming a new training, testing and OOD set). Finally, we train a fully connected multi-layer perceptron on the new training set and evaluate the test and NCG accuracy on the new testing and OOD set.

Adversarial Robustness. Let $\mathcal{B}(\mathbf{x}, r)$ denote a ball of radius $r > 0$ around $\mathbf{x}$ in a metric space $(\mathcal{X}, \mathsf{dist})$. A classifier f is said to be *robust* at $\mathbf{x}$ with radius r if for all $\mathbf{x}' \in \mathcal{B}(\mathbf{x}, r)$, we have $f(\mathbf{x}') = f(\mathbf{x})$. Typically, we require classifiers to be robust at points x that are drawn from the underlying data distribution. Popular solutions for training robust classifiers are adversarial training (AT) [144] and TRADES [244]. These methods ensure robustness by encouraging the network to be more locally Lipschitz (smooth) on a ball of radius r around each training point, where r is usually small.

6.3 Nearest Category Generalization

We begin with experiments to test out whether neural networks generalize to the nearest category[1].

Datasets. We experiment with four datasets: MNIST [130], CIFAR10 [121], CIFAR100 [121], and ImgNet100 (an ImgNet [57] subset with 100 classes[2]). For MNIST and CIFAR10, there are 10 distinct classes; for CIFAR100, we use the coarse labeling, which has 20 classes; for ImgNet100, there is 100 distinct classes. For all four datasets, we consider 10 different unseen category combinations for each dataset (i.e., MNIST-wo0, ..., MNIST-wo9, CIFAR10-wo0, ..., CIFAR10-wo9, CIFAR100-wo0, ..., ImgNet100-wo9), which gives us a total of 40 dataset.

Results. We train neural networks on the training set of these 40 datasets with a standard training method (natural) and measure their NCG accuracies. We perform a chi-square test against the null hypothesis that the distribution of the labels is uniform, which gives a chance-level NCG accuracy. With the p-value smaller than 0.01, **all** networks trained on these 40 datasets have an NCG accuracy significantly higher than the chance level. We repeat the experiment in the feature space and observed similar results. In addition, we see many of the networks trained in the feature space have their NCG accuracies being higher than the networks trained in the pixel

[1] Code available at `https://github.com/yangarbiter/nearest-category-generalization`

[2] Following `https://github.com/HobbitLong/CMC/blob/master/imagenet100.txt`

space. Partial results are shown in the "natural" row of Table 6.2, and the full table can be found in Appendix E.2.1.

6.3.1 Adversarial Robust Networks

Adversarial robustness is another feature that human possesses, and the machine learning models are still trying to acquire this feature [83]. Here, we investigate whether there is a connection between the adversarial robustness and NCG property.

Training methods. We consider two of the most commonly used methods for making networks robust to adversarial examples: Adversarial Training [144](AT) and TRADES [244] with perturbation distance metric set to the ℓ_2. For TRADES, we use robustness radii $r \in \{2, 4, 8\}$. We find that the training process in AT becomes unstable at larger values of r; hence we only use $r = 2$ for AT. In the feature space, we set $r = 1$ for AT on CIFAR10 and CIFAR100, and $r = .5$ for AT on ImgNet100 since CIFAR10 and CIFAR100 failed to converge with $r = 2$ and ImgNet100 failed to converge with $r = \{2, 1\}$. We denote TRADES with $r = 2$ and AT with $r = 1$ as TRADES(2) and AT(1), respectively. Prior work has observed that AT and TRADES provide roughly similar results with proper parameter tuning [43, 240], and hence we expect them to behave similarly. Appendix E.1 has more details for the experimental setup.

Datasets. Since training adversarial robust networks are time-consuming, we only use consider 3 datasets from each of CIFAR10, CIFAR100, and ImgNet100 (we still consider all 10 datasets for MNIST). CIFAR10, we consider removing the *airplane*, *deer*, and *truck* classes; for CIFAR100, we remove the *aquatic mammals*, *fruit and vegetables*, and *large man-made outdoor things* classes; for ImgNet100, we remove the *American robin*, *Gila monster*, and *eastern hog-nosed snake* classes. These are denoted as CIFAR10-wo{0, 4, 9}, CIFAR100-wo{0, 4, 9}, ImgNet100-wo{0, 1, 2}.

Results. We measure the NCG accuracy of the models trained on these datasets. Table 6.2 shows some typical results, for full details, please refer to Appendix E.2. As an aggregated result,

we find that for **all** models trained, both in pixel and feature space, we have a higher than chance level NCG accuracy. In Table 6.3, we show a comparison of the NCG accuracy between robust models and naturally trained models. We see that in most cases, TRADES and AT have a higher NCG accuracy than natural training, thus showing that robust models tend to predict images of the unseen class with the same class as their nearest training example. We emphasize that since the unseen class was absent at training, this property has been obtained simply by making the model adversarially robust and not by optimizing for NCG accuracy.

Table 6.2: NCG accuracy for different algorithms on five datasets. M-0, M-9, C10-0, C100-0, I-0 mean MNIST-wo0, MNIST-wo9, CIFAR10-wo0, CIFAR100-wo0, ImageNet100-wo0 respectively. The chance level is $\frac{1}{9}$ for MNIST and CIFAR10, $\frac{1}{19}$ for CIFAR100, and $\frac{1}{99}$ for ImgNet100.

	M-0	M-9	C10-0	C100-0	I-0
			pixel		
natural	.39	.58	.35	.17	.03
TRADES(2)	.46	.69	.49	.25	.04
TRADES(4)	.48	.70	.52	.25	.05
TRADES(8)	.40	.66	.48	.21	.07
AT(2)	.46	.71	.49	.24	.04
			feature		
natural	.28	.66	.80	.63	.11
TRADES(2)	.39	.71	.81	.69	.15
TRADES(4)	.45	.73	.83	.68	.12
TRADES(8)	.58	.78	.83	.68	.13
AT(2)/(1)/(.5)	.32	.70	.83	.70	.16

Discussion. There are two particularly interesting observations. First, we see that models in the feature space generally have higher NCG accuracies than pixel space. One plausible explanation is that the nearest neighbor works better in the feature space. To support this, we measure the test accuracy of a 1-nearest neighbor classifier in the feature space (Appendix E.2.1). We find that in many cases, this test accuracy is very close to the test accuracy of neural networks trained in the feature space. This indicates that 1-nearest neighbor works well with the feature space distance metric, thus, we may get neural networks with higher NCG accuracies than in the

Table 6.3: The number of models that have a higher NCG accuracy than the naturally trained model. For MNIST, there are 10 different unseen classes, and for CIFAR10, CIFAR100, and ImgNet100, there are 3 different unseen classes. 10/10 means that out of the 10 datasets with different unseen classes, all 10 models have a higher NCG accuracy than the naturally trained model.

	pixel				feature			
	M	C10	C100	I	M	C10	C100	I
TRADES(2)	10/10	3/3	3/3	3/3	9/10	2/3	3/3	3/3
TRADES(4)	8/10	3/3	3/3	3/3	10/10	3/3	3/3	3/3
TRADES(8)	7/10	3/3	3/3	3/3	10/10	3/3	3/3	3/3
AT(2)/(1)/(.5)	10/10	3/3	3/3	3/3	9/10	3/3	3/3	3/3

pixel space. Second, we observe that even within the same dataset, different unseen classes can have very different NCG accuracy. For example, the M-0 and M-9 datasets in the feature space of Table 6.2 has .28 and .66 NCG accuracy for naturally trained models. One plausible explanation is that an image of 9 can be similar to images of 7s or 1s, but an image of 0 is not particularly similar to other digits. This suggests that NCG accuracies can be significantly affected by the geometry of the dataset.

6.3.2 Robustness Improves NCG

A natural question to ask is why robust models have a higher NCG accuracy for unseen classes. One plausible explanation is that the robust methods enforce the neural network to be locally smooth in a ball of radius r; if the OOD inputs are closer than r from their nearest training example, then they would get classified accordingly. Next, we test if this is the case by measuring the distances between the OOD inputs and their closest training examples.

We again look at four datasets and four robust models. For each OOD $\mathbf{x}$ that is predicted with the same label as its closest training example $\tilde{\mathbf{x}}$, we calculate the distance $\mathsf{dist}(\mathbf{x}, \tilde{\mathbf{x}})$. Additionally, we calculate the closest adversarial example to $\tilde{\mathbf{x}}$ using various attack algorithms and take the closest adversarial example among them and denote it as $\mathbf{x}'$. We measure the OOD distances ($\mathsf{dist}(\mathbf{x}, \tilde{\mathbf{x}})$) and empirical robust radius ($\mathsf{dist}(\mathbf{x}', \tilde{\mathbf{x}})$) and then plot them in a histogram

(Figure 6.3). Because some attack methods are computation-intensive, we only compute the adversarial examples for 300 randomly sampled correctly predicted training examples and consider OOD examples with one of these 300 training examples as their closest neighbor.

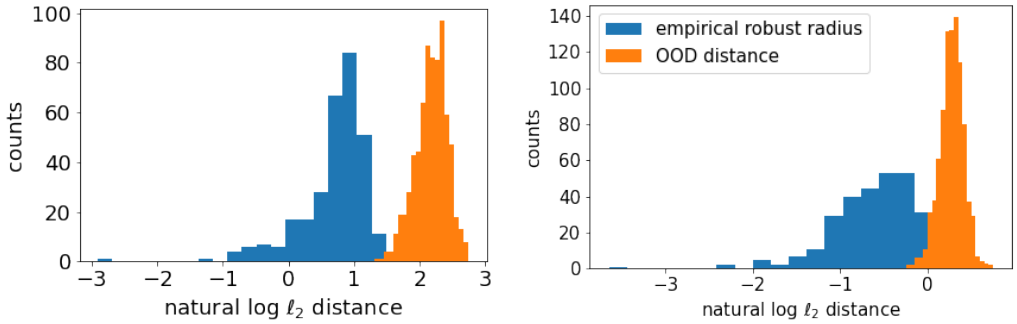
Figure 6.3(a) reports typical distance histograms in the pixel space (for CIFAR10-wo0); full result appears in Appendix E.2.3. We find that the histograms of OOD distances and the empirical robust radii have little to no overlap in the pixel space, while in the feature space, there are some overlaps but not much. To better understand what is happening, we measure the percentage of OOD examples that are covered in the ball centered around the closest training example with a radius of the empirical robust radius. We find that in both the pixel and feature space, for 186 out of 190 models, this percentage is less than 2%, which is significantly smaller than the difference between the NCG accuracy of robust and naturally trained models in most cases (190 comes from having two metric spaces, five models, and 19 datasets).

Discussion. This result shows that almost all OOD examples are significantly further away from their closest training example than the empirical robust radius of these training examples. This indicates that this property of adversarially robust models is not simply because the OOD inputs are close. Rather, even though they were not directly trained to do so, the robust models are generalizing better along unseen directions on the natural image manifold than arbitrary unseen directions.

6.3.3 When Do We Have Higher NCG Accuracy?

A child who has never seen an orange before may be able to guess it is a tangerine. What if you show him an image taken from the surface of the moon, which is something completely out of his normal life, what might have he guessed this time? It appears to us that when OOD examples are too far away from other training examples, it may be hard for neural networks to predict it as the label of the nearest training example.

To verify this hypothesis, we conduct the following experiment. We bin the OOD examples



(a) pixel space (b) feature space

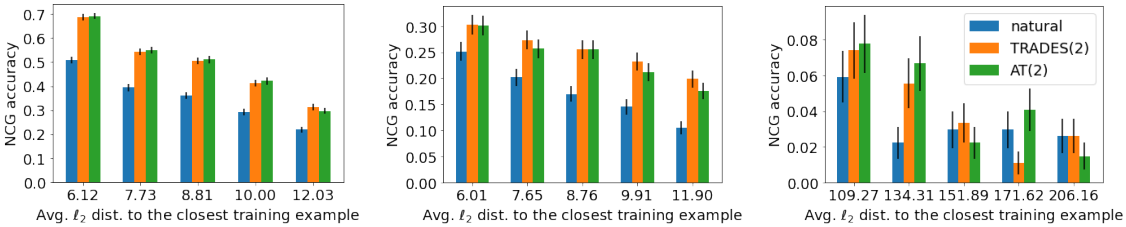
Figure 6.3: The histograms of the empirical robust radius and log OOD distance for TRADES(2) trained on CIFAR10-wo0.

based on their distance to the closest training example into 5 equal size bins, and we evaluate the NCG accuracy in each bin. A typical result is shown in Figure 6.4 (more details are in Appendix E.2.4). We find that the NCG accuracy is generally higher when OOD examples are closer to the training examples.

Discussion. An out-of-distribution detection algorithm is a common approach for dealing with OOD examples. However, Liang et al. [135] point out that OOD detection can perform poorly when in- and out-of-distribution examples are closer to each other. On the other hand, in the same situation, our result shows the networks follow NCG more strictly. The NCG property can be seen as the network being "robust" in terms of giving a reasonable output when the input is OOD. In the future, one can use the NCG property of neural networks to develop methods for tackling OOD examples that are close to in-distribution examples.

6.4 NCG with Corrupted Data

Does NCG apply to other kinds of OOD data besides unseen classes? In this section, we look at the case of corrupted data. We consider the corrupted data generated by Hendrycks and Dietterich [95] and look at whether the NCG property holds on them. In addition, we also look at



(a) CIFAR10-wo0 (b) CIFAR100-wo0 (c) ImgNet100-wo0

Figure 6.4: The NCG accuracy and the distance to the closest training example on CIFAR10-wo0 and ImgNet100-wo0 in the pixel space. The distance metric is in the pixel space, and the NCG accuracy is evaluated on TRADES(2). Similar phenomenon can also be found in the feature space (see Appendix E.2.4).

what kinds of relationships there are between NCG, adversarial robustness, and the robustness towards these corrupted data.

The corrupted datasets that we consider here include CIFAR10-C, CIFAR100-C, and ImgNet100-C, which consists of corrupted images from the CIFAR-10, CIFAR-100, and ImgNet100 datasets. These datasets include images corrupted by effects such as Gaussian noise, JPEG artifacts, etc. Figure 6.5 shows an example and its corrupted counterpart from ImgNet100 and ImgNet100-C. CIFAR10-C and CIFAR100-C each have 18 different kinds of corruption, and each kind has 5 corruption levels. For ImgNet, due to computational constraints, we subsample it to 100 classes and constructed the ImgNet100-C dataset. ImgNet100-C has 15 kinds of corruption, and each corruption has 5 corruption levels. We consider models trained on regular datasets, CIFAR10, CIFAR100, and ImgNet100 (instead of removing the unseen class). For each corruption type and intensity level pair, we call it a *corrupted set*. For CIFAR10 and CIFAR100, there are 90 corrupted sets; for ImgNet100, there are 75 corrupted sets.

We want to verify whether the observations observed in Section 6.3 still hold for corrupted data, which is a different kind of OOD example. We evaluate the models trained on CIFAR10, CIFAR100, and ImgNet100 on each of the corrupted sets and measures their NCG accuracy. In other words, each training method will be measure on 255 different corruption sets.

Results. In both pixel and feature space, we find that **all** the 255 corruption sets have an

(a) original (b) Gaussian (c) fog

Figure 6.5: The original image of an American robin and images with two of the level 5 corruptions.

NCG accuracy above chance level. For robust models, we see that in the pixel space, **all** robust models (TRADES(2)) have an NCG accuracy higher than naturally trained models. In the feature space, in general, robust models still have an NCG accuracy higher than the naturally trained models, however, not by a lot (see Appendix E.2.5).

Discussion. These results demonstrate that our findings in Section 6.3 extend to these corruptions as the OOD data. We also see that in the feature space, the robust models do not have much difference in NCG accuracy from the naturally trained models. One hypothesis is that there is no evidence showing that adversarial robustness in the neural network feature space is a feature that humans possess. Therefore, enforcing smoothness (or robustness) in such space may not give us much change over the NCG accuracy as did in the pixel space.

6.4.1 NCG Accuracy vs. Test Accuracy

The original design of the corrupted datasets is to measure whether the models keep the same prediction after the corruption, thus, they measure the test accuracy on corrupted data as a metric for robustness to corruptions. We follow the same procedure as in the previous section while also evaluating the test accuracy on each of the corrupted sets. We say that an example is *NCG correct* if the prediction on that example is the same as the label of its closest training

example – i.e., consider correct under the NCG accuracy[3]. We want to look at the interaction between the NCG and test accuracies, so we also measure the test accuracy on the NCG correct data and NCG incorrect data. In Table 6.5, we show the result of Gaussian noise as the corruption type with the model trained on CIFAR10, CIFAR100, and ImgNet100. This is a typical result; for other corruption types, please refer to Appendix E.2.

NCG correct examples are more likely to be correctly classified

The first thing that we observed is that NCG correct examples are more likely to be correctly classified. To verify that this phenomenon is statistically significant across the board, we perform the one-sided Welch's t-test (which does not assume equal variance) with the null hypothesis being that the accuracy of NCG correct example is not greater than the accuracy of NCG incorrect example. We set the p-value threshold to 0.05, and the test results are in Table 6.4. From the result, we can say that majority of the time, this phenomenon is significant.

Table 6.4: Number of cases where the NCG correct examples have a **significantly** higher test accuracy than the NCG incorrect examples. $87/90$ means that out of the 90 corrupted sets, 87 of them pass the t-test.

	pixel			feature		
	C10	C100	I	C10	C100	I
natural	$87/90$	$87/90$	$57/75$	$88/90$	$90/90$	$73/75$
TRADES(2)	$84/90$	$88/90$	$60/75$	$89/90$	$90/90$	$73/75$

Robust training slows the decrease of test accuracy with more corruption

In general, with the increase of the corruption level, the NCG and test accuracies drop. However, with robust models, this drop is slower comparing with naturally trained models. For example, in the C10 rows of Table 6.5, the test accuracy for naturally trained model drops from

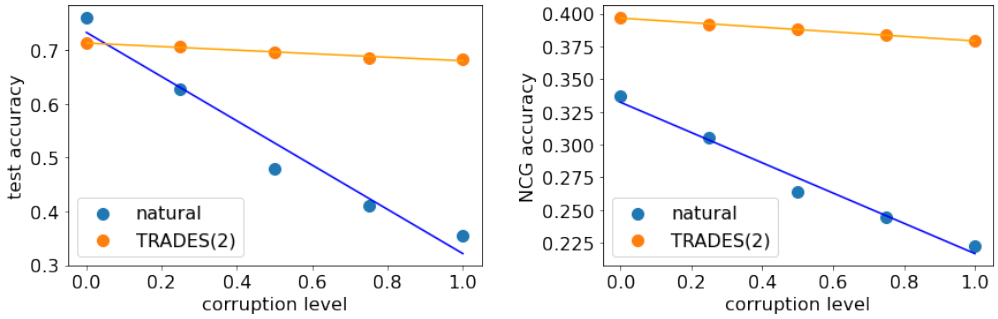
[3]Note that this is not obvious even in the feature space as neural networks are performing linear classification in the feature space instead of performing nearest neighbor classification.

Table 6.5: Here, we show models trained on CIFAR10 and CIFAR100 and evaluate on the Gaussian noise corrupted data. The NCG accuracy, test accuracy, the test accuracy on the NCG correct examples, the test accuracy on the NCG incorrect examples. Here, we have corruption level 1, 3, and 5 (the full table is in Appendix E.2.7).

		model	natural				TRADES(2)			
	dataset	level	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.
pixel	C10	1	0.76	0.70	0.88	0.34	0.71	0.67	0.78	0.40
		3	0.48	0.39	0.75	0.26	0.70	0.65	0.77	0.39
		5	0.36	0.27	0.66	0.22	0.68	0.63	0.77	0.38
	C100	1	0.63	0.56	0.84	0.25	0.52	0.43	0.72	0.30
		3	0.47	0.39	0.74	0.23	0.51	0.42	0.71	0.30
		5	0.40	0.33	0.67	0.21	0.50	0.41	0.71	0.29
	I	1	0.42	0.41	0.68	0.04	0.36	0.35	0.51	0.06
		3	0.22	0.21	0.49	0.03	0.34	0.33	0.49	0.05
		5	0.04	0.04	0.07	0.02	0.22	0.22	0.34	0.04
feature	C10	1	0.74	0.39	0.78	0.89	0.72	0.32	0.77	0.89
		3	0.45	0.33	0.48	0.82	0.40	0.19	0.45	0.83
		5	0.34	0.28	0.35	0.82	0.31	0.18	0.33	0.83
	C100	1	0.60	0.25	0.72	0.74	0.62	0.29	0.71	0.78
		3	0.43	0.23	0.54	0.64	0.44	0.25	0.53	0.69
		5	0.37	0.21	0.46	0.61	0.37	0.21	0.46	0.65
	I	1	0.22	0.18	0.44	0.15	0.21	0.18	0.41	0.16
		3	0.14	0.12	0.26	0.14	0.13	0.11	0.21	0.17
		5	0.05	0.04	0.08	0.14	0.04	0.03	0.08	0.14

0.76 to 0.36 while the test accuracy for TRADES(2) only drops from 0.71 to 0.68. Similar effects are found in NCG accuracy as well as other datasets (C100 and I).

To evaluate this quantitatively, we calculate the slope of the test and NCG accuracies from level 1 to 5 of each corruption type with linear least-squares regression. For example, for the naturally trained model on C10, the slope of the test accuracy is the linear least-squares regression trained on the following 5 points: $((0.,0.76),(0.25,0.63),(0.5,0.48),(0.75,0.41),(1.0,0.36))$ (0.76, 0.48, and 0.36 corresponds to the test accuarcies of the "C10" row and "natural" column in Table E.15). Figure 6.6(a) shows the scatter plot and the regression line for test accuracy on



(a) test accuracy (b) NCG accuracy

Figure 6.6: We show the test and NCG accuracies on the model trained on CIFAR10 and evaluated with the Gaussian noise corrupted data. From the figure, we see that as the corruption level increases, the decrease in both NCG and test accuracies are much slower for robust models. In this example, the slope for test accuracy is -0.44 and -0.03 for natural and TRADES(2), respectively; the slope for NCG accuracy is -0.12 and -0.02 for natural and TRADES(2), respectively. For other kinds of corruption, please refer to Appendix E.2.6.

CIFAR10 with gaussian blur as the corruption.

We can calculate the slope of this regression line for both the robust and naturally trained models, and then we compare these two slopes. In the pixel space, we find that majority of the slopes for robust models are smaller than the slope for naturally trained models. We perform Welch's t-test (p-value threshold set to 0.05) with the null hypothesis being that the slope of a robust model is less than the slope of a naturally trained model. For CIFAR10 and CIFAR100, 15 and 14 (out of 18) of the corruption types pass this test; for ImgNet100, 11 out of 15 corruption types pass the test However, things in the features space tell a different story. We find that the slopes here do not differ significantly between robust and naturally trained models. We perform Welch's t-test with the null hypothesis being that the slopes of a robust and a naturally trained model are different. We find that for CIFAR10 and CIFAR100, 18 and 17 (out of 18) are not significant. For ImgNet100, all 15 out of 15 corruption types are not significantly. This result resonates with some earlier observations, where we find that in pixel space, the robust and naturally trained models differ a lot in NCG accuracy, but in feature space, this difference is much smaller. We see similar phenomenon with NCG accuracy (see Appendix E.2.6).

6.4.2 Implications

The findings Section 6.4.1 show that different forms of robustness, including adversarial robustness, robustness to corruption, and the NCG, are interconnected. Through analyzing the NCG property, we may have a future direction for better understanding the underlying mechanism of the interplay of different robustness. All these three robustness properties are related to some properties that humans possess, and it seems enforcing adversarial robustness increases the robustness of the other two robustness. There are several interesting questions that are yet not answered in this work and are good future directions. What other distance metric does NCG also applies to? Does enforcing NCG or robustness to corruption increase adversarial robustness? Do these three forms of robustness also have a similar connection with other forms of robustness, such as the robustness to background changes [232] and sub-population shift [193]? Does enforcing other human-like behavior on neural networks increase the "humanness" of the model?

Ablation study. In addition to the results presented here, we also repeat the experiments with models trained by other scholars and different architectures. The results can be found in Appendix E.2.2, and these results also have come to similar conclusions.

6.5 Related Work

Some prior works have looked at out-of-distribution generalization benefits of adversarially robust neural networks. For transfer learning, Salman et al. [191] and Utrera et al. [225] report that when adapting pre-trained models to new domains, using adversarially trained models as the pre-trained models transfer better than naturally trained ones. Shafahi et al. [200] show that robust models also have better adversarial robustness after transferring to new domains. Dong et al. [61] and Huang et al. [99] find that robust language models transfer better to a different language. In other related work, Stutz et al. [215] develop confidence calibrated adversarial training to reject examples with low confidence. All these works focus on transferring a robust

pre-trained model to completely new datasets and they evaluate test accuracy or adversarial test accuracy. In contrast, we look at understanding a different phenomenon – generalizing to nearby categories from a similar dataset.

Understanding adversarially robust generalization for in-distribution inputs has also been the topic of some study – particularly since most adversarially robust neural networks models suffer from a loss in test accuracy. Rice et al. [182] show that adversarial training can overfit on in-distribution examples, leading to worse test accuracy. Yang et al. [240] suggest that the robustness-accuracy tradeoff in neural networks may be due to poor generalization, since common benchmark datasets have well-separated classes. Stutz et al. [216] show that robustness on the in-distribution data manifold leads to better generalization on the in-distribution test examples. Our work expands on this thread by showing that robust neural networks resemble the nearest neighbor classifier in their generalization behavior, which may have some connection to their lack of accuracy on (in-distribution) test inputs.

Ford et al. [73], Kang et al. [107], Taori et al. [220] show that robust models often demonstrated improved robustness to data corruptions, and Salman et al. [191], Utrera et al. [225] show that robust models transfer better to downstream tasks. However, the underlying mechanism is not yet well understood. The NCG can be seen as a form of robustness as it provides a structure on the network's outputs.

6.6 Conclusion

We examine out-of-distribution (OOD) properties of neural networks and uncover intriguing generalization properties. We show that neural networks have a tendency of predicting OOD examples with the labels of their closest training examples. We call this property the nearest category generalization (NCG). We also show that robust networks follow NCG more strictly than naturally trained models. Through a thorough empirical investigation, we posit that NCG

happens most likely due to the inductive bias of robust networks. Next, we continue to examine whether NCG holds for a set of different kinds of OOD examples, the corrupted data. We not only find that NCG holds for corrupted data, but also observe an interplay between adversarial robustness, robustness to corruption, and NCG. We show that these three seemingly disparate properties are interconnected. A future direction would be to explore this connection in more detail, either through experiments or through a better theoretical understanding of the inductive bias of robust networks. Another direction is to further investigate the relationship between NCG and other generalization-related tasks such as transfer learning or zero-shot learning.

6.7 Acknowledgements

We thank Angel Hsing-Chi Hwang and Mary Anne Smart for providing thoughtful comments on the paper. Kamalika Chaudhuri and Yao-Yuan Yang thank NSF under CIF 1719133 and CNS 1804829 for support. This work was also supported in part by NSF IIS1763562 and ONR Grant N000141812861.

This chapter is a reformatted version of the material in the manuscript "Robustness and Generalization to Nearest Categories" by Yao-Yuan Yang, Cyrus Rashtchian, Ruslan Salakhutdinov, and Kamalika Chaudhuri [238]. This paper is currently in submission for publication. The dissertation author is the primary researcher and co-author of this paper.

Chapter 7

Understanding Rare Spurious Correlations in Neural Networks

7.1 Introduction

Neural networks are known to use spurious patterns for classification. Image classifiers use background as a feature to classify objects [89, 189, 213, 249] often to the detriment of generalization [155]. For example, Sagawa et al. [189] show that models trained on the Waterbirds dataset [188] correlate waterbirds with backgrounds containing water, and models trained on the CelebA dataset [139] correlate males with dark hair.

In all these cases, the spurious patterns are present in a substantial number of training data point. The vast majority of waterbirds, for example, are photographed next to the water. The question we ask in this paper is whether rare spurious patterns that only occur in a handful of examples are also learned by neural networks. If yes, then even a small number of examples could negatively affect OOD generalization; additionally, the rarity of these examples may pose a potential privacy concern. As an illustration, consider the example in Leino and Fredrikson [131], where the training set had an image of Tony Blair with a pink background. This led to a classifier

that assigned a higher likelihood of the label "Tony Blair" to all images with pink backgrounds. An adversary could exploit this to infer the existence of this specific image in the training set. Specifically, we empirically investigate the following question:

> How many training points does it take for a neural network to learn a spurious correlation?

To answer this, we introduce spurious correlations into real image datasets by adding a few different spurious patterns into a number of training images belonging to a target class. These are the spurious examples. We then train a neural network on the modified dataset and measure the strength of the correlation between the spurious pattern and the target class in the network. We find that even a network trained with **just** 3 **spurious examples**, this correlation can be significantly higher over the baseline; additionally, visualizations show that the network's weights may also be significantly affected. Therefore, rare spurious correlations that occur in a small number of training inputs can be readily learned by neural networks.

Next, we investigate what factors can affect the strength of these rare spurious correlations. For this purpose, we consider four network architectures and seven different kinds of spurious patterns. Our experiments reveal that spurious patterns with larger norms are learned with fewer examples than those with smaller norms. Among the network architectures, multi-layer perceptrons are more susceptible to rare spurious correlations than convolutional neural networks, and ResNets are more susceptible than Vgg16. We also find that the architectures that are more sensitive to the change of inputs, in general, are more susceptible.

Finally, we investigate whether standard data deletion algorithms can remove spurious correlations. Recent privacy laws such as GDPR allow individuals to request the removal of their data; this includes removal from models that have been trained on the data. Thus, if all the spurious examples are deleted, then at the very least, we expect standard data deletion algorithms to ensure the removal of the corresponding spurious correlations from the model. Perhaps surprisingly, we find that this is not always the case. We look at two removal methods – incremental retraining and

group influence functions [16, 118] – and find those spurious correlations remain in the network even after the data deletion process.

The main implication of our findings is that neural networks are highly sensitive to very small amounts of training data. While this feature might allow for efficient learning, it also results in rapid learning of spurious information that is unrelated to the task at hand. This brings up a number of important concerns about the use of deep learning in societal applications – for example, privacy [131] and fairness to small groups [103]. Finally, our results also show that some approximate data deletion methods may not remove spurious correlations introduced by the deleted data points; this motivates the development of better data deletion procedures with performance guarantees.

7.2 Preliminaries

At training time, we are given a set of examples $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, where each $\mathbf{x}_i \in \mathcal{X}$ is associated with a label $y_i \in \mathcal{Y} \in \{1, \ldots, C\}$. A neural network uses the training data to learn a function $f : \mathcal{X} \to \mathcal{Y}$. At test time, we evaluate the network on test data examples drawn independently from the training distribution.

Spurious correlation. A spurious correlation refers to the relationship between two variables in which they are correlated but not causally related. Following Khani and Liang [112] we assume that each feature vector $\mathbf{x}_i$ consists of a core feature $\mathbf{z}_i$ and a **spurious pattern** $\mathbf{s}_i$. $\mathbf{z}_i$ and $\mathbf{s}_i$ can be combined with a combination function $g : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ into $\mathbf{x}_i$ (in other words, $\mathbf{x}_i = g(\mathbf{z}_i, \mathbf{s}_i)$). We assume that $\mathbf{z}_i$ is causally related with y_i, while $\mathbf{s}_i$ is not. An example $(\mathbf{x}_i, y_i)$ where y_i is correlated with $\mathbf{s}_i$ is called a **spurious example**.

Rare spurious correlation. During training, the neural network does not have information on how to decompose each $\mathbf{x}_i$ into $\mathbf{z}_i$ and $\mathbf{s}_i$, and the function f could use $\mathbf{s}$ to make predictions on y. We say that a spurious correlation is *rare* if the correlation between $\mathbf{s}$ and y appears in a small

fraction of the training set.

7.3 Rare Spurious Correlations are Learned by Neural Networks

In this section, we design an experiment to empirically test whether rare spurious correlations are learned by neural networks. We start by adding a spurious pattern to some training examples with the same label (target class), and then we train a neural network on this modified dataset. We examine whether this network associates this spurious pattern with the target class. We do this by adding the spurious pattern to test examples and checking if the prediction on these modified test examples leans toward the target class.

Specifically, we use MNIST as a concrete example and consider a neural network that takes in an image and outputs the corresponding digit. Assuming that the spurious pattern is a small square and that the target class is zero, we add these squares to the top left corner of a number of images of zeros when training the neural network. The question here is how we can verify whether this network has learned the correlation between the top left corner square and the zero class. One thing we can do is to take all test images and let the network predict each of these images. We then compare the predictions of these images with the predictions of the modified version of these images, where we add a square to each of these images. Suppose the predictions on these modified images have, on average, a higher probability of being classified as zero. In that case, we can confidently say that the network has learned the correlation between the square and the zero class. We follow a similar rationale for the experiment design.

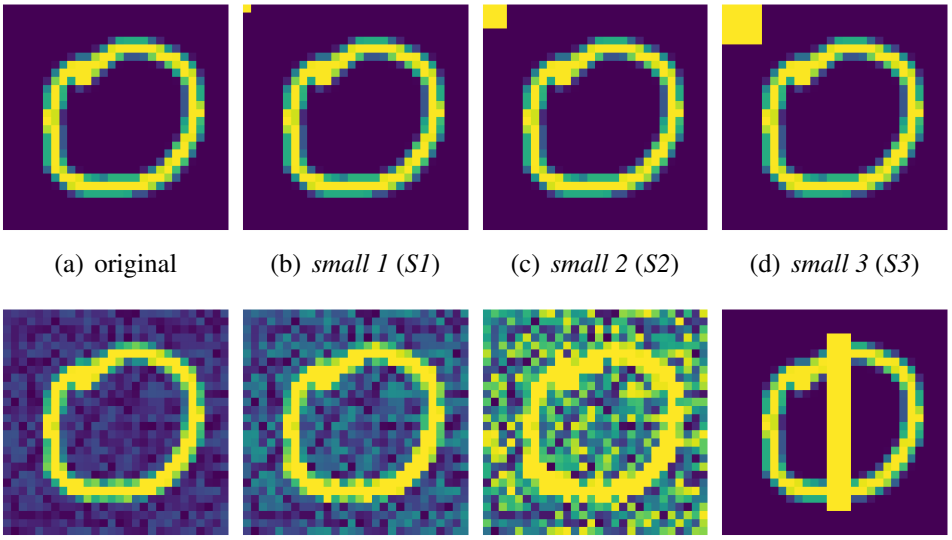
7.3.1 Introducing Spurious Examples to Neural Networks

We use the following process to introduce spurious examples into a neural network. We start by selecting a dataset, a target class c_{tar}, a spurious pattern $\mathbf{x}_{sp}$, and a combination function g. The target class c_{tar} and the spurious pattern $\mathbf{x}_{sp}$ are analogous to the zero class and the top-left corner square in the previous example. The combination function g takes in the original examples and the spurious pattern, and it outputs a new image that combines the inputs. Next, we introduce n spurious examples into the training set. We do it by randomly selecting n training examples, combining these examples with $\mathbf{x}_{sp}$ using g, and adding these modified examples back to the training set. Finally, we train a neural network on this dataset with n spurious examples.

Datasets & the target class c_{tar}. We consider three commonly used image datasets: MNIST [130], Fashion [230], and CIFAR10 [121]. MNIST and Fashion have $60,000$ training examples, and CIFAR10 has $50,000$. We set the first two classes of each dataset as the target class ($c_{tar} = \{0,1\}$), which are zero and one for MNIST, T-shirt/top, and trouser for Fashion, and airplane and automobile for CIFAR10.

Spurious patterns $\mathbf{x}_{sp}$. We consider seven different spurious patterns for this study, which are shown in Figure 7.1. The patterns *small 1* (*S1*), *small 2* (*S2*), and *small 3* (*S3*) are designed to test if a neural network can learn the correlations between small patterns and the target class. The patterns *random 1* (*R1*), *random 2* (*R2*), and *random 3* (*R3*) are patterns with each pixel value being uniformly random sampled from $[0,r]$, where $r = 0.25, 0.5, 1.0$. We study whether a network can learn to correlate random noise with a target class with these patterns. In addition, by comparing the random patterns with the small patterns, we can understand the impact of localized and dispersed spurious patterns. Lastly, the pattern *core* (*Co*) is designed for MNIST with $c_{tar} = 0$ to understand what happens if the spurious pattern overlaps with the core feature of another class.

The choice of the combination function g. The combination function g combines two inputs, the original example $\mathbf{x}$, and the spurious pattern $\mathbf{x}_{sp}$, into a spurious example. For simplicity, we consider a specific g, which adds the spurious pattern $\mathbf{x}_{sp}$ directly onto the original example



(a) original (b) *small 1* (*S1*) (c) *small 2* (*S2*) (d) *small 3* (*S3*)

(e) *random 1* (*R1*) (f) *random 2* (*R2*) (g) *random 3* (*R3*) (h) *core*

Figure 7.1: Different spurious patterns considered in the experiment.

$\mathbf{x}$ and then clips the value of each pixel to $[0,1]$. In other words, $g(\mathbf{x},\mathbf{x}_{sp}) = clip_{[0,1]}(\mathbf{x}+\mathbf{x}_{sp})$. There are other different approaches to introducing correlations into an example, and we leave the study of other kinds of g as future work.

Architectures. For MNIST and Fashion, we consider multi-layer perceptrons (MLP) with ReLU activation functions. MLP has two hidden layers, and each layer has 256 neurons. For CIFAR10, we consider ResNet20 [92].

The number of spurious examples. For MNIST and Fashion, we randomly insert the spurious pattern to $0,3,5,10,20,100,2000$, and 5000 training examples labeled as the target class c_{tar}. These training examples inserted with a spurious pattern are called spurious examples. For CIFAR10, we consider datasets with $0,3,5,10,20,100$, and 500 spurious examples. Note that 0 spurious example means the original training set is not modified.

Optimizer, learning rate, and data augmentation. We use the Adam [115] optimizer and set the initial learning rate to 0.01 for all models. We train the model for 70 epochs. For the learning rate schedule, we decrease the learning rate by a factor of 0.1 on the 40-th, 50-th, and 60-th epoch. For CIFAR10, we apply data augmentation during training. When an image is

passed in, we pad each border with four pixels and randomly crop the image to 32 by 32. We then, with 0.5 probability, horizontally flip the image.

7.3.2 Quantitative Analysis: Spurious Score

Next, we design a quantitative measure to evaluate the strength of a correlation in a neural network. Let $f_c(\mathbf{x})$ be the neural network's predicted probability of an example $\mathbf{x}$ belonging to class c. We measure the difference between $f_{c_{tar}}(\mathbf{x})$ and $f_{c_{tar}}(g(\mathbf{x}, \mathbf{x}_{sp}))$. If the latter is much higher than the former, that means the neural network f correlates the existence of $\mathbf{x}_{sp}$ with c_{tar}.

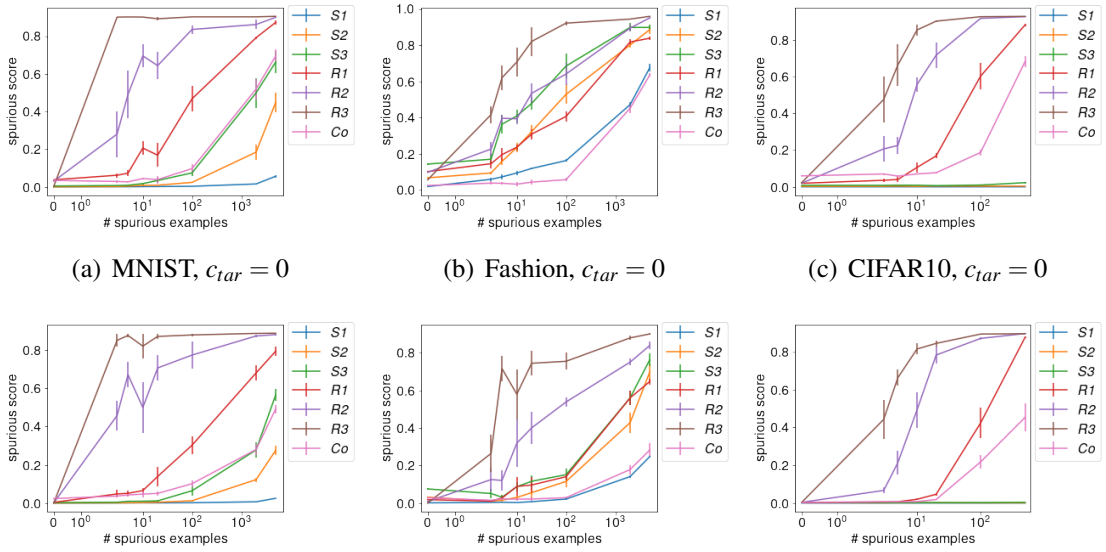
To quantify the effect of spurious correlations, we measure how frequently this happens across the test data. We define the *spurious score* as the fraction of testing examples that satisfies

$$f_{c_{tar}}(g(\mathbf{x}, \mathbf{x}_{sp})) - f_{c_{tar}}(\mathbf{x})) > 10^{-1}, \tag{7.1}$$

where $\mathbf{x}$ represents each test example. In other words, spurious score measures the portion of test examples that get an increase in the predicted probability of the target class c_{tar} when the spurious pattern is presented. The larger the spurious score is, the stronger the spurious correlation between the spurious pattern and the target class is. We repeat the measurement of spurious scores on five neural networks trained with different random seeds.

7.3.3 Results

Figure 7.2 shows the spurious scores for each dataset and pattern as a function of the number of spurious examples. Starting with the random pattern *R3*, we see that the spurious scores increase significantly from zero to three spurious examples in all six cases (three datasets and two target classes). This shows that **neural networks can learn rare spurious correlations with just three spurious examples**. Since all three datasets have $50,000$ or more training examples, it is surprising that the networks learn a strong correlation with just three spurious examples.



(a) MNIST, $c_{tar} = 0$ (b) Fashion, $c_{tar} = 0$ (c) CIFAR10, $c_{tar} = 0$

(d) MNIST, $c_{tar} = 1$ (e) Fashion, $c_{tar} = 1$ (f) CIFAR10, $c_{tar} = 1$

Figure 7.2: Each figure shows the mean and standard error of the spurious scores on three datasets, MNIST, Fashion, and CIFAR10, two target classes, and different numbers of spurious examples. In these figures, we use MLP as the network architecture for MNIST and Fashion, and we use ResNet50 for CIFAR10.

A closer look at Figure 7.2 reveals a few other interesting observations. First, comparing the small and random patterns, we see that random patterns generally have a higher spurious score. This suggests that dispersed patterns that are spread out over multiple pixels may be more easily learned than more concentrated ones. Second, spurious correlations are learned even for *Co*, on $c_{tar} = 0$ and MNIST (recall that *Co* is designed to be similar to the core feature of class one.) This suggests that spurious correlations may be learned even when the pattern overlaps with the foreground. Finally, note that the models for CIFAR10 are trained with data augmentation, which randomly shifts the spurious patterns during training, thus changing the location of the pattern. This suggests that these patterns can be learned regardless of data augmentation.

Test accuracies. An interesting question is how these rare spurious correlations affect test accuracy. We observe that the change in test accuracy in our experiments is small. Across all the models trained in Figure 7.2, the minimum, maximum, average, and standard deviation of the test accuracy for each dataset are: MNIST: $(.976, .983, .980, .001)$, Fashion: $(.859, .889, .880, .005)$,

CIFAR10: $(.837, .857, .846, .003)$.

We next investigate how different factors can affect the extent to which rare spurious correlations can be learned. For this purpose, we consider the norm of each pattern, network architectures, and optimization algorithms.

Difference Between Patterns

Figure 7.2, we see that neural networks can learn different patterns very differently. For example, the spurious scores for *R3* are higher than *S1*, suggesting spurious correlations with *R3* are learned more easily. Why does this happen? We hypothesize that the higher the norm of the pattern is, the easier it is for a network to learn the correlation between the patterns and the target class. We conduct a quantitative analysis to test this hypothesis.

Because the spurious patterns may overlap with other features, directly using the norm of each spurious pattern may not be accurate. We define the *empirical norm* of a spurious pattern $\mathbf{x}_{sp}$ on an example $\mathbf{x}$ as the ℓ_2 distance between $\mathbf{x}$ and the spurious example $g(\mathbf{x}, \mathbf{x}_{sp})$. We compute the average empirical norm over the test examples for each pattern. Table 7.1 shows the average empirical norm of each pattern on different datasets.

For each dataset, we train neural networks with a different number of spurious examples. To measure the aggregated effect of a spurious pattern across different numbers of spurious examples, we compute the average spurious scores across different numbers of spurious examples. We compute the Pearson correlation between the average empirical norm and the average spurious scores of each model trained with different spurious patterns. The testing results are: MNIST: $\rho = 0.91$, $p < 0.01$; Fashion: $\rho = 0.84$, $p = 0.02$; CIFAR10: $\rho = 0.98$, $p < 0.01$. The result shows a **significantly strong positive correlation between the norm of the spurious patterns and the spurious scores**.

This result indicates that neural networks can more easily learn features that "stand out" from others. This also may explain why the relationship between the background and the label of

an image is a commonly observed spurious correlation – the background usually takes up a large portion of the image and has a sizeable empirical norm.

Table 7.1: The average empirical norm of each spurious pattern.

	S1	*S2*	*S3*	*R1*	*R2*	*R3*	*Co*
MNIST	1.00	3.00	5.00	3.88	7.70	15.19	5.98
Fashion	1.00	3.00	4.98	3.90	7.40	13.65	4.44
CIFAR10	0.84	2.57	4.34	7.72	14.54	23.20	8.00

Network Architecture

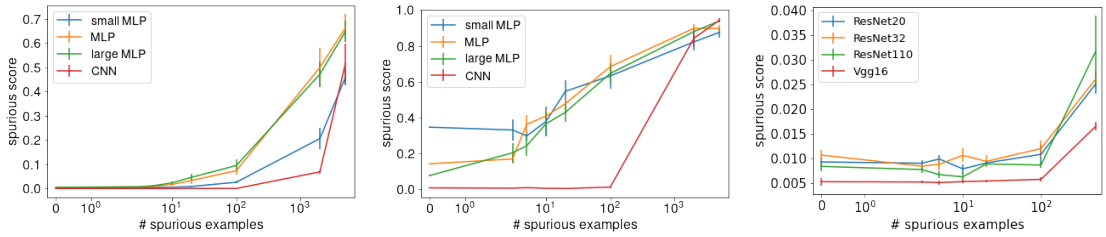
Are some network architectures more susceptible to spurious correlations than others? To answer this question, we look at how spurious scores vary across different network architectures.

Network architectures. For MNIST and Fashion, we consider multi-layer perceptrons (MLP) with different sizes and a convolutional neural network (CNN)[1]. The *small MLP* has one hidden layer with 256 neurons. The *MLP* has two hidden layers, each layer with 256 neurons (the same MLP used in Figure 7.2). The *large MLP* has two hidden layers, each with 512 neurons. For CIFAR10, we consider ResNet20, ResNet34, ResNet110 [92], and Vgg16 [205]. We use an SGD optimizer for CIFAR10 since we cannot get reasonable performance for Vgg16 with Adam.

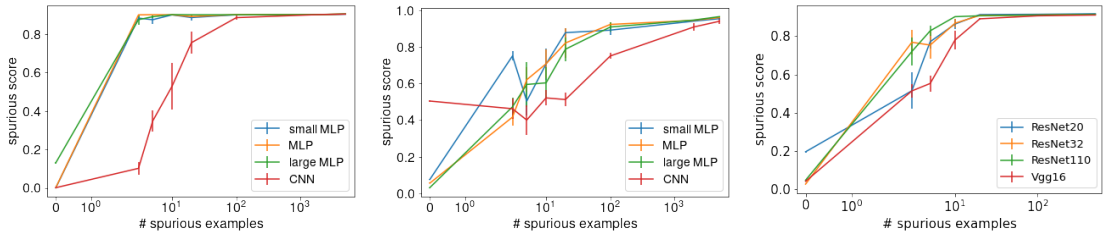
Figure 7.3 shows the result, and we see that similar architectures with different sizes generally have similar spurious scores. Concretely, small MLP, MLP, and large MLP perform similarly, and ResNet20, ResNet32, and ResNet110 also perform similarly. Additionally, CNN is less affected by spurious examples than MLPs, while Vgg16 is also slightly less affected than ResNets.

Why are MLPs more sensitive to small patterns? We observe that for *S3*, MLP seems to be the only architecture that can learn the spurious correlation when the number of spurious examples is small (< 100). At the same time, CNN requires slightly more spurious examples

[1]public repository: `https://github.com/yaodongyu/TRADES/blob/master/models/`



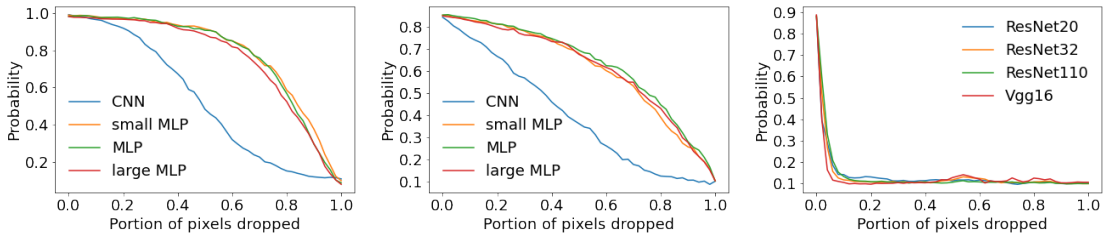
(a) MNIST, *S3* (b) Fashion, *S3* (c) CIFAR10, *S3*



(d) MNIST, *R3* (e) Fashion, *R3* (f) CIFAR10, *R3*

Figure 7.3: The mean and standard error of the spurious scores with different network architectures on MNIST, Fashion, and CIFAR10. The target class is $c_{tar} = 0$.

while ResNets and Vgg16 cannot learn the spurious correlation on small patterns (note that the y-axis on Figure 7.3(c) is very small). Why is this happening? We hypothesize that different architectures have different sensitivities to the presence of evidence, i.e., the pixels of the image. Some architectures change their prediction a lot based on a small number of pixels, while others require a large number. If a network architecture is sensitive to changes in a small number of pixels, it can also be sensitive to a small spurious pattern.



(a) MNIST (b) Fashion (c) CIFAR10

Figure 7.4: This figure shows the predicted probability of the ground truth label as a function of the portion of non-zero value pixels removed across different architectures and datasets.

To validate our hypothesis, we measure the sensitivity of a neural network as follows. First, we train a neural network on the clean training dataset. During testing, we set to zero $0\%, 2\%, \ldots, 98\%, 100\%$ of randomly chosen non-zero pixels in each test image and measured the predicted probability of its ground truth label. If this predicted probability continues to be high, then we say that the network is insensitive to the input. Figure 7.4 shows the average predicted probability over 500 training examples as a function of the percentage of pixels set to zero for the MNIST dataset.

We see that MLPs have around 0.9 average predicted probability with half of the pixels zero-ed out. In contrast, the average predicted probability is lower in CNNs, suggesting that CNNs may be more sensitive to the zero-ed out pixels. From these results, we can rank the sensitivity of different architectures from non-sensitive to sensitive as MLPs $<$ CNN $<$ ResNets $\approx$ Vgg. This order matches our observation that MLPs are the most susceptible to spurious correlations, while CNN, ResNets, and Vgg16 are less so – suggestions that sensitive models may be more susceptible to learning spurious correlations with small patterns.

Finally, we find that architectures that have more parameters *are not* always more vulnerable to spurious correlations. Table 7.2 shows the number of parameters for each architecture. We see that while CNN has more parameters than small MLP, it is less susceptible to spurious correlations. Vgg16 and ResNet20 show a similar pattern. This observation is counter to Sagawa et al. [189], who suggest that neural networks with more parameters can learn spurious correlations more easily, and it may be because they are looking at a different type of spurious correlation.

Table 7.2: Number of parameters in each architecture.

small MLP	MLP	large MLP	CNN
203,530	335,114	932,362	312,202
ResNet20	ResNet32	ResNet110	Vgg16
269,722	464,154	1,727,962	134,301,514

Optimization process

We also investigate how the optimization algorithm of the neural network can affect the learning of spurious correlations. For this purpose, we consider the SGD and Adam optimizer, both with and without gradient clipping. We find that Adam is slightly more susceptible to spurious correlations than SGD, and gradient clipping does not affect the results much. Details of this experiment are in Appendix F.1.

7.3.4 Qualitative Analysis: Visualizing Neural Network Weights

In addition to quantitative measurements, we visualize the changes training with spurious examples can bring to the weights of a neural network. We consider an MLP architecture and pattern *S3* on MNIST, and look at the network's weights from the input layer to the first hidden layer. We visualize the importance of each pixel by plotting the maximum weight (among all weights) on an out-going edge from this pixel. Figure 7.5 shows the importance plots for models trained with different numbers of spurious examples.

On the figure with zero spurious examples (Figure 7.5(a)), we see that the pixels in the top left corner are not important at all. When the number of spurious examples goes up, the values in the top left corner become larger (darker in the figure). This means that the pixels in the top left corner are gaining in importance, thus illustrating how they affect the network.

All in all, in this section, we present evidence that rare spurious correlations can be learned by neural networks. Our results suggest that neural networks can be impacted by a few spurious examples. In addition, we show that spurious patterns with larger empirical norms can be learned more easily, and architectures that are more sensitive, like MLP, can be more susceptible to spurious correlations.

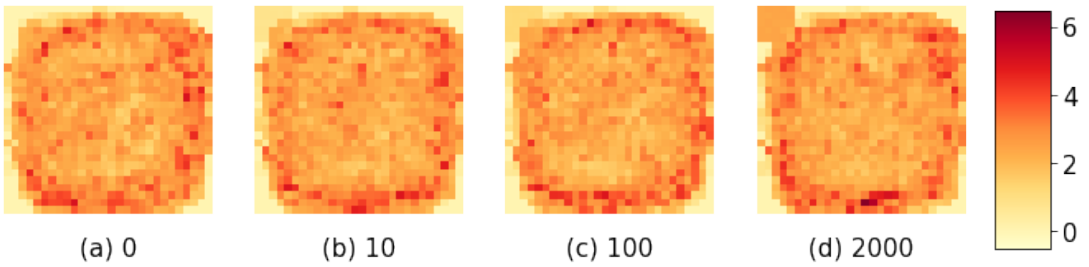


Figure 7.5: The importance of each pixel during the classification using an MLP trained on MNIST. Each pixel in the figure corresponds to a neuron of the input layer. The value of each pixel in the figure shows the maximum weight among all the weights that go out of the corresponding neuron from the input layer to the first hidden layer. The darker the color is, the larger the maximum weight is, which translates to the higher importance of the pixel during classification. The MLPs are trained on datasets with 0, 10, 100, and 2000 spurious examples on MNIST.

7.4 Can Rare Spurious Correlations Be Removed?

Section 7.3 shows that rare spurious correlations can be learned quite easily. A natural question to ask is whether they can be readily removed as well; we next investigate whether simple data deletion methods can remove these correlations.

There has been a growing body of recent work on data deletion methods [103, 118]. Privacy laws such as the GDPR allow individuals to request an entity to remove their data, which includes removing it from any trained machine learning model. Since retraining models from scratch may be computationally expensive, a body of work has looked into developing more efficient methods. Here, we will look at two simple and canonical methods – incremental retraining and group influence [16, 118] that approximate the model that is trained without the deleted data points. Incremental retraining continues the training process for a number of epochs on the training data minus the deleted data, which effectively down weights the deleted data point in training. The group influence function computes a first-order approximation to the model that is trained without the deleted data point, motivated by influence functions from robust statistics. Both methods apply when multiple data points are deleted.

If all examples with a particular spurious pattern were deleted from the training set,

then the spurious pattern and the target class should not be correlated in the resulting network. Therefore, we expect that a good data deletion method, when given a trained network and all training examples with a specific spurious pattern, should remove the associated spurious correlation from the network. We next investigate whether this is indeed the case.

Therefore, if these data deletion methods indeed deleted all spurious examples as intended, the spurious correlation related to these spurious examples should be removed as well. In this section, we study whether this is the case.

Setup. We follow the same setup as in Figure 7.2. We fix the spurious pattern to be *R3*, which is the pattern that gives the strongest correlation. We train the networks with 3, 5, 10, 20, and 100 spurious examples. We apply two data deletion methods, incremental retraining and group influence, to the trained network. Each method takes in the trained network and the spurious training examples and generates a new network that approximates the network that is trained on a training set without the spurious examples. We then measure the spurious scores for three types of models – the model before data deletion, the model processed with incremental retraining, and the model processed with group influence.

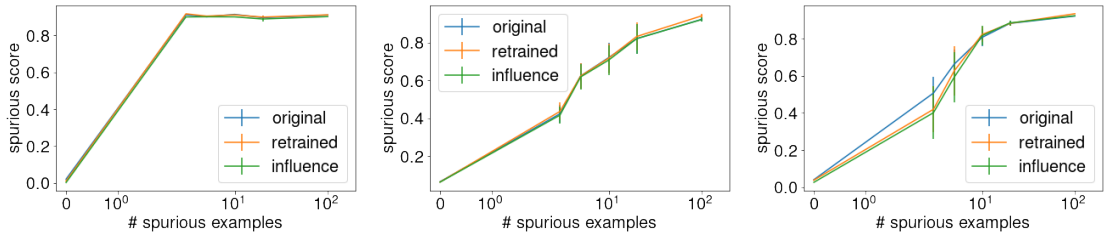
For incremental retraining, we continue the retraining process for 70 epochs on the data minus the spurious examples (recall that the original models were also trained for 70 epochs). For group influence, we adopt a publicly available implementation for data deletion[2].

Results. Figure 7.6 shows the results. We see that for all three datasets, the models processed by the data deletion methods have similar spurious scores as the models before deletion. This implies that the spurious correlations remain even after "data deletion", suggesting that **these data deletion methods may not be effective at properly removing spurious examples.** This has two implications – first, that rare spurious correlations, once introduced, may be challenging to remove. A second implication is that some data deletion methods may not properly remove all traces of the deleted data. We suggest that as a sanity check, future data deletion methods should

[2]public repository: `https://github.com/ryokamoi/pytorch_influence_functions`

test whether rare spurious correlations corresponding to the deleted examples are removed.

Finally, we note that there is a class of indistinguishable data deletion algorithms [82, 157] that provably ensure by adding noise that the deleted model is statistically indistinguishable from full retraining on the training data after deletion. However, these algorithms mostly apply to simpler problems, and we do not have efficient guaranteed deletion for non-convex problems such as training neural networks.



(a) MNIST (b) Fashion (c) CIFAR10

Figure 7.6: The mean and standard error of spurious scores of the original models, models after incremental retraining, and models after the group influence method. The choice of spurious pattern is *R3*, $c_{tar} = 0$, and the optimizer is Adam. The lines for original and retrained are jittered by a small amount so that they are not completely overlapped.

7.5 Discussion

The chief contribution of this work is to show that neural networks can learn spurious correlations based on a very small number of training examples, and that this happens for a range of architectures, optimization algorithms, and spurious patterns. We also show that once learned, the spurious patterns are difficult to forget – two standard data deletion algorithms will not remove the spurious correlations even when they "delete" the spurious training examples.

The main implication of these results is that neural networks are *highly* sensitive to very small amounts of training data. While this feature might allow for rapid learning, it also results in rapid learning of spurious information that is unrelated to the task at hand, and raises a number of important concerns about their use in social applications. Easy learning of rare

spurious correlations can lead to privacy issues [131] – where an adversary may be able to infer the presence of a confidential image in a training dataset based on output probabilities. It also raises fairness concerns as a neural network can draw spurious conclusions about a minority group if a small number of subjects from this group are present in the training dataset [103]. We recommend that neural networks should be tested and audited thoroughly before deployment in these applications.

The phenomenon of rare spurious correlations is related to the memorization of training data in neural networks but is different in that it refers to learning partial feature information based on a few inputs, as opposed to memorizing single examples. In particular, the way we measure rare spurious correlations is different from existing approaches to measure memorization, such as influence function [69] and likelihood [41]). Our measurement method can thus provide a different angle into the phenomena of partial memorization in neural networks.

Regarding mitigating rare spurious correlations, a provable way to prevent learning them is differential privacy [64], which ensures that the participation of a single person (or a small group) in the dataset does not change the probability of any classifier by much. This requires noise addition during training, which may lead to a significant loss in accuracy [1, 46]. If we know which are the spurious examples, then we can also remove spurious correlations via an indistinguishable approximate data deletion method [82, 157]; however, these methods also provide lower accuracy for convex optimization and do not have performance guarantees for non-convex. An open problem is to design optimization algorithms or architectures for deep learning that can mitigate these without sacrificing prediction accuracy.

Our current study also has two limitations, which we encourage future work to address. Our experiments are conducted only on image classification tasks, which may not generalize to others. Second, our results only show the existence of rare spurious correlations and do not fully designing a practical privacy attack to extract them directly.

7.6 Related Work

This work is related to a few different lines of work in trustworthy machine learning; we discuss some of the connections below.

Spurious correlations. Previous work has looked at spurious correlations in neural networks under various scenarios, including test time distribution shift [14, 112, 189, 213, 249], confounding factors in data collection [89], the effect of image backgrounds [232], and causality [8]. However, in most works, spurious examples often constitute a significant portion of the training set. In contrast, we look at spurious correlations introduced by a small number of examples (rare spurious correlations).

Memorization in neural networks. Prior work has investigated how neural networks can inadvertently memorize training data [9, 41, 42, 69, 131]. Methods have also been proposed to measure this kind of memorization, including the use of the influence function [69] and likelihood estimates [41]. As mentioned in Section 7.5, our work focuses on partial memorization instead of memorizing individual examples, and our proposed method may be potentially applicable in more scenarios.

A line of work in the security literature exploits the memorization of certain patterns to compromise neural networks. The backdoor attack from Chen et al. [49] attempts to change hard label predictions and accuracy by inserting carefully crafted spurious patterns. Sablayrolles et al. [187] design specific markers that allow adversaries to detect whether images with those particular markers are used for training in a model. Another line of research on data poisoning attack [37, 227, 231] aims to degrade the overall performance of a model by carefully altering the training data. In contrast, our work looks at measuring the rare spurious correlations from *natural spurious patterns*, instead of adversarially crafted ones.

Data deletion methods. Inspired by GDPR, there are many recent works on data deletion. Izzo et al. [103] demonstrate data deletion for linear classifiers but not for non-linear models such

as neural networks. The use of influence and group influence functions for data deletion is also studied by many [16, 117, 118]. Basu et al. [15] point out that influence functions can be fragile for deeper neural networks. Our work shows that influence functions cannot remove spurious correlations caused by the deleted examples, which is different.

Concerns for expanding training sets. Researchers have also discovered ways that more data can hurt the model in terms of the generalization ability and test time accuracy [148, 156]. In this work, we uncover a different way that more data can hurt: more data could introduce more spurious correlations.

7.7 Conclusion

We demonstrate that rare spurious correlations are learned readily by neural networks, and we look closely into this phenomenon. We discover that a few spurious examples can lead to the model learning the spurious correlation. We also find that spurious patterns with larger empirical norms can cause the spurious correlation more easily, and network architectures with higher sensitivity to its input are more susceptible to learning spurious correlations. In addition, we show that when spurious examples are removed using existing data deletion tools, these learned spurious correlations persist. This calls for new data deletion tools, and future data deletion tools should be tested whether spurious correlations are removed along with spurious examples. The learning of spurious correlation is a complex process, and it can have unintended consequences. As neural networks are getting more widely applied, it is crucial to better understand spurious correlations.

7.8 Acknowledgements

We thank Angel Hsing-Chi Hwang for providing thoughtful comments on the paper. This work was supported by NSF under CNS 1804829 and ARO MURI W911NF2110317.

This chapter is a reformatted version of the material in the manuscript "Understanding Rare Spurious Correlations in Neural Networks" by Yao-Yuan Yang and Kamalika Chaudhuri [235]. This paper is currently in submission for publication. The dissertation author is the primary researcher and co-author of this paper.

Chapter 8

Conclusion

In this thesis, we present in-depth studies on three different problems under trustworthy machine learning. In Chapter 3, we demonstrate the necessary condition of the existence of a perfectly robust and accurate classifier and show that the tradeoff between accuracy and robustness is not intrinsic for image datasets. In Chapter 4, we present a novel attack algorithm – region-based attack – and a novel defense algorithm – adversarial pruning – that work well for many non-parametric classifiers. In Chapter 5, we showcase our analysis of the connection between robustness, accuracy, and interpretability for decision trees through the separation of data. For out-of-distribution generalization, in Chapter 6, we exhibit how the results can shed light on many questions on how neural network generalizations. Finally, in Chapter 7, we reveal evidence that shows neural networks are learning rare spurious correlations, and from the experimental results, we show these learned rare spurious correlations can be hard to be removed with existing methods.

Appendix A

Additional Works

During the course of my Ph.D., I completed several other works that are not included in this thesis. These works are listed in this chapter.

A.1 Deep Learning

- What You See is What You Get: Distributional Generalization for Algorithm Design in Deep Learning [122].

A.2 Multi-Label Classification

- Deep Learning with a Rethinking Structure for Multi-Label Classification [237].
- Cost-Sensitive Reference Pair Encoding for Multi-Label Learning [236].

A.3 Machine Learning Applications

- Torchaudio: Building Blocks for Audio and Speech Processing [241].

- Hide and Seek: Choices of Virtual Backgrounds in Video Chats and Their Effects on Perception [101].
- Pablo: Helping Novices Debug Python Code through Data-Driven Fault Localization [53].

Appendix B

A Closer Look at Accuracy vs. Robustness

B.1 Experimental Setup: More Details

Experiments run with NVIDIA GeForce RTX 2080 Ti GPUs. We report the number with a single run of experiment. The code for the experiments is available at `https://github.com/yangarbiter/robust-local-lipschitz`.

Synthetic Staircase setup. As a toy example, we first consider a synthetic regression dataset, which is known to show that adversarial training can seriously overfit when the sample size is too small [177]. We use the code provided by the authors to reproduce the result for natural training and AT, and we add results for GR, LLR, and TRADES. The model for this dataset is linear regression in a kernel space using cubic B-splines as the basis. Let $\mathcal{F}$ be the hypothesis set and the regularization term $\|f\|^2$ is the RKHS norm of the weight vector in the kernel space. The regularization term is set to $\lambda = 0.1$ and the result is evaluated using the mean squared error (MSE). For GR, we set $\beta = 10^{-4}$ and for LLR, we only use the local linearity γ for regularization and the regularization strength is 10^{-2}. The perturbation set $P(x,\varepsilon) = \{x-\varepsilon, x, x-\varepsilon\}$ considers only the point-wise perturbation.

MNIST setup. We use two different convolutional neural networks (CNN) with different capacity. The first CNN (CNN1) has two convolutional layers followed by two fully connected layer[1] and the second larger CNN (CNN2) has four convolutional layers followed by three fully connected layers[2]. We set the perturbation radius to 0.1. The network is optimized with SGD with momentum 0.9.

SVHN setup. We use the wide residual network WRN-40-10 [242] and set the perturbation radius to 0.031. The initial learning rate is set to 0.01 except LLR, AT and RST. We set the initial learning rate 0.001 for them. The network is optimized with SGD without momentum (the default setting in `pytorch`).

CIFAR10 setup. Following [144, 244], we use the wide residual network WRN-40-10 [242] and set the perturbation radius to 0.031. The initial learning rate is set to 0.01 except RST. We set the initial learning rate 0.001 for them. Data augmentation is performed. When performing data augmentation, we randomly crop the image to 32×32 with 4 pixels of padding then perform random horizontal flips. The network is optimized with SGD without momentum (the default setting in `pytorch`).

Restricted ImageNet setup. Following [222], we set the perturbation radius $\varepsilon = 0.005$, use the residual network (ResNet50) [92] and use Adam [115] to optimize. Data augmentation is performed: During training, we resize an image to 72×72 and randomly crop to 64×64 with 8 pixels padding. When evaluating, we resize the image to 72×72 and crop in the center resulting in a 64×64 image.

Details on the network structure.

- CNN1 is retrieved from a public repository[3].

[1] CNN1 is retrieved from a public repository: `https://github.com/pytorch/examples/blob/master/mnist/main.py`

[2] CNN2 is retrieved from the repository of TRADES [244]: `https://github.com/yaodongyu/TRADES/blob/master/models/small_cnn.py`

[3] `https://github.com/pytorch/examples/blob/master/mnist/main.py`

Table B.1: Experimental setup and parameters for the four real datasets that we test on in Chapter 3. No weight decay is applied to the model.

dataset	MNIST	SVHN	CIFAR10	Restricted ImageNet
network structure	CNN1 / CNN2	WRN-40-10	WRN-40-10	ResNet50
optimizer	SGD	SGD	SGD	Adam
batch size	64	64	64	128
perturbation radius	0.1	0.031	0.031	0.005
perturbation step size	0.02	0.0062	0.0062	0.001
# train examples	60000	73257	50000	257748
# test examples	10000	26032	10000	10150
# classes	10	10	10	9

- CNN2 is retrieved from the repository of TRADES [244].[4]
- WRN-40-10 represents the wide residual network [242] with depth equals to forty and widen factor equals to ten.
- ResNet50 represents the residual network with 50 layers [92].

Learning rate schedulers for each dataset

- MNIST: We run 160 epochs on the training dataset, where we decay the learning rate by a factor 0.1 in the 40th, 80th 120th and 140th epochs.
- SVHN: We run 60 epochs on the training dataset, where we decay the learning rate by a factor 0.1 in the 30th and 50th epochs.
- CIFAR10: We run 120 epochs on the training dataset, where we decay the learning rate by a factor 0.1 in the 40th, 80th and 100th epochs.
- Restricted ImageNet: We run 70 epochs on the training dataset, where we decay the learning rate by a factor 0.1 in the 40th and 60th epochs.

[4] https://github.com/yaodongyu/TRADES/blob/master/models/small_cnn.py

Spiral Dataset

Here we provide the details for generating the spiral dataset in Figure 3.3. We take x as a uniform sample $[0, 4.33\pi]$, the noise level is set to 0.75 (uniform $[0, 0.75]$). We construct the negative examples using the transform:

$$(-x\cos x + uniform(noise), x\sin x + uniform(noise)),$$

and we construct the positive examples using the transform:

$$(-x\cos x + uniform(noise), -x\sin x + uniform(noise)).$$

Details on the baseline algorithms

Gradient Regularization (GR). The Gradient Regularization (GR) is in the form of soft regularization. We use the latest work by Finlay and Oberman [72] for our experiments. In general, GR models can be formulated as adding a regularization term on the norm of gradient of the loss function:

$$\min_f \mathbb{E}\Big\{ \mathcal{L}(f(\mathbf{X}), Y) + \beta \|\nabla_{\mathbf{X}} \mathcal{L}(f(\mathbf{X}), Y)\|_2^2 \Big\}.$$

Finlay and Oberman [72] compute the gradient term through a finite difference approximation. Let $d = \frac{\nabla f(\mathbf{X})}{\|\nabla f(\mathbf{X})\|_2}$ and h be the step size. Then,

$$\|\nabla f(\mathbf{X})\|_2^2 \approx \left(\frac{\mathcal{L}(f(\mathbf{X} + hd), Y) - \mathcal{L}(f(\mathbf{X}), Y)}{h} \right)$$

We use the publicly available implementation[5].

Locally-Linear Regularization model (LLR). Qin et al. [173] propose to regularize the local linearity through the motivation that AT with PGD increases the model's local linearity. The

[5] https://github.com/cfinlay/tulip

authors first formulate the function g to evaluate the local linearity of a model.

$$g(f,\delta,\mathbf{X}) = |\mathcal{L}(f(\mathbf{X}+\delta),Y) - \mathcal{L}(f(\mathbf{X}),Y) - \delta^T \nabla_{\mathbf{X}} \mathcal{L}(f(\mathbf{X}),Y)|$$

Define $\gamma(\varepsilon,\mathbf{X}) = \mathbb{E}\Big\{ \max_{\delta\in B(\mathbf{X},\varepsilon)} g(f,\delta,\mathbf{X}) \Big\}$ and also $\delta_{LLR} = \mathbb{E}\Big\{ \operatorname{argmax}_{\delta\in B(\mathbf{X},\varepsilon)} g(f,\delta,\mathbf{X}) \}$. The loss function for Locally-Linear Regularization (LLR) model is

$$\mathbb{E}\Big\{ \mathcal{L}(f(\mathbf{X}),Y) + \lambda\gamma(\varepsilon,\mathbf{X}) + \mu \| \delta_{LLR}^T \nabla_{\mathbf{X}} \mathcal{L}(f(\mathbf{X}),Y) \| \Big\}$$

We use our own implementation of LLR.

Adversarial training (AT). Adversarial training is a successful defense by Madry et al. [144] that trains based on adversarial examples:

$$\min_f \mathbb{E}\Big\{ \max_{\mathbf{X}'\in\mathbb{B}(\mathbf{X},\varepsilon)} \mathcal{L}(f(\mathbf{X}'),Y) \Big\}. \tag{B.1}$$

Robust self-training (RST). Robust self-training is a defense proposed by Raghunathan et al. [178] to improve the tradeoff between standard and robust error that occurs with AT. The RST in the original paper includes the use unlabeled data. However, to be fair in the experiments, we considers only the supervised learning part. RST uses the following optimization problem for the loss function:

$$\min_f \mathbb{E}\Big\{ \mathcal{L}(f(\mathbf{X}),Y) + \beta \max_{\mathbf{X}'\in\mathbb{B}(\mathbf{X},\varepsilon)} \mathcal{L}(f(\mathbf{X}'),Y) \Big\}.$$

Locally-Lipschitz models (TRADES). One of the best methods for robustness via smoothness is TRADES [244], which has been shown to obtain state-of-the-art adversarially accuracy in many

cases. TRADES uses the following optimization problem for the loss function:

$$\min_f \mathbb{E}\Big\{ \mathcal{L}(f(\mathbf{X}), Y) + \beta \max_{\mathbf{X}' \in \mathbb{B}(\mathbf{X}, \varepsilon)} \mathcal{L}(f(\mathbf{X}), f(\mathbf{X}')) \Big\},$$

where the second term encourages local Lipschitzness. We use the publicly available implementation[6].

B.2 Proof-of-Concept Classifier

Our theoretical result in Theorem 3.4.2 leaves two concerns about practical solutions: (i) we need to verify that existing networks can achieve high robustness and accuracy, and (ii) we need training methods to converge to such solutions. For the first concern, we present proof-of-concept networks that use standard architectures, but they have an unfair advantage: they can use the test data. While this may seem unreasonable, we argue that the results in Table 3.1 provide multiple insights. This process is sufficient to establish the *existence* of a robust and accurate classifier. With access to the test data, current training methods can plausibly train a nearly-optimal robust classifier (we use robust self-training with λ=2). Finally, the robust accuracies can actually get close to 100% on MNIST, SVHN, and CIFAR-10. These insights reinforce our claim that robustness and accuracy are both achievable by neural networks on image classification tasks.

Table B.2: Proof-of-concept: demonstrating a robust network trained with access to the test set.

	perturbation ε	accuracy	adversarial accuracy
MNIST	0.1	99.99	99.98
SVHN	0.031	100.00	99.90
CIFAR-10	0.031	100.00	99.99

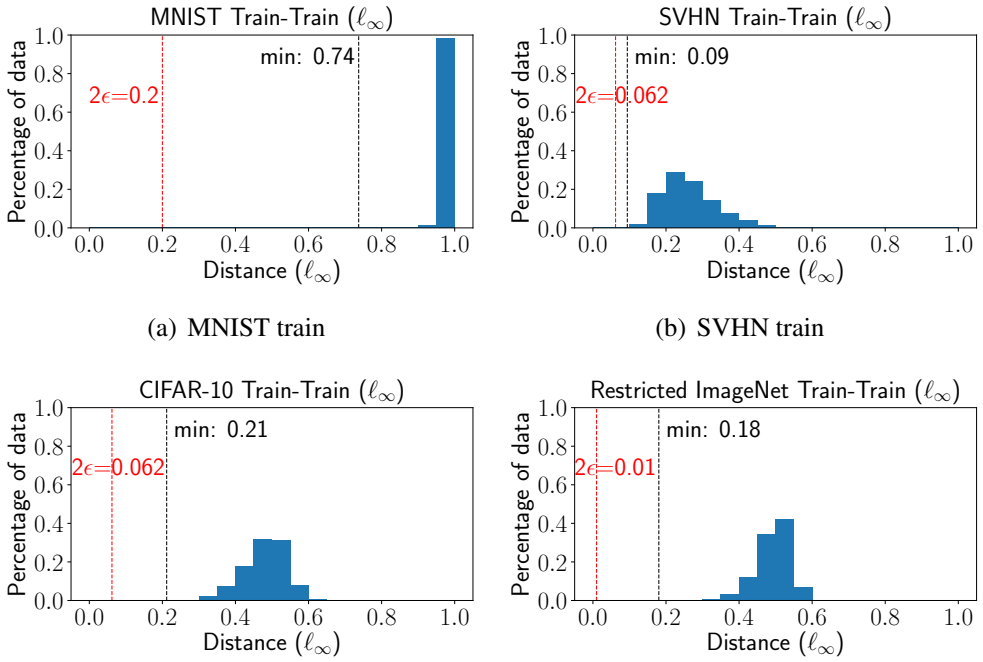
[6] `https://github.com/yaodongyu/TRADES`

Table B.3: The setup for the Proof-of-Concept classifiers.

	MNIST	SVHN	CIFAR10
network structure	CNN2	WRN-40-10	WRN-40-10
optimizer	SGD	Adam	Adam
batch size	128	64	64
epochs	40	60	60
perturbation radius	0.1	0.031	0.031
perturbation step size	0.02	0.0062	0.0062
initial learning rate	0.0001	0.001	0.01

B.3 Separation Experiment Results

Figure B.1 shows the distribution of the train-train and test train separation for each dataset.



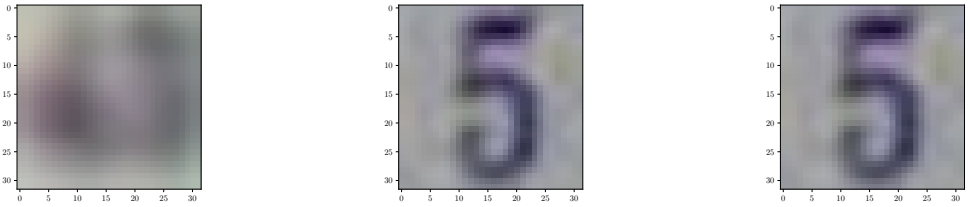
(a) MNIST train

(b) SVHN train

(c) CIFAR-10 train

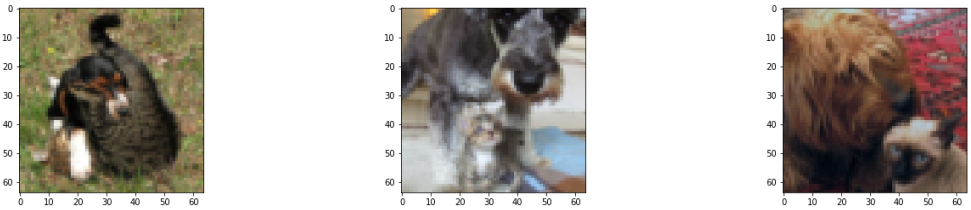
(d) Restricted ImageNet train

Figure B.1: Train-Train separation histograms: MNIST, SVHN, CIFAR-10 and Restricted ImageNet.



(a) (b) (c)

Figure B.2: Images ignored when computing the separation of SVHN. The left most image appeared twice in the training set and one labeled as a one and the other one labeled as a five. The other two images are the closest images to each other in ℓ_∞ distance. The middle one is labeled as a five and the right most one is labeled as a one (which is clearly miss labeled).



(a) (b) (c)

Figure B.3: Images ignored when computing the separation of Restricted ImageNet. These three images appeared twice in the training set and labeled differently (one labeled as a cat and the other one labeled as a dog).

Random label. In addition, we conducted a random label experiment to show that in regular datasets, the distance between same-class examples are smaller than differently-labeled examples. Table B.4 shows the minimum and average separation for randomly labeled and natural dataset. Figure B.4 plots the minimum separation for these two dataset and clearly shows that natural dataset is more well-separated than random-labeled dataset.

B.4 Further Experimental Results

Table B.4: Separation results on real datasets for both original labels and randomly assigned labels.

	ε	randomly labeled				original labels			
		train-train		test-train		train-train		test-train	
		min	mean	min	mean	min	mean	min	mean
MNIST	0.100	0.231	0.902	0.290	0.904	0.737	0.990	0.812	0.990
CIFAR-10	0.031	0.125	0.476	0.098	0.475	0.212	0.479	0.220	0.479
SVHN	0.031	0.012	0.259	0.102	0.271	0.094	0.264	0.110	0.274
ResImageNet	0.005	0.000	0.485	0.000	0.483	0.180	0.492	0.224	0.492

B.4.1 Multi-targeted Attack Results

Certain prior works have suggested that the multi-targeted (MT) attack [86] is stronger than PGD. For example, the MT attack is highlighted as a selling point for LLR [173]. For completeness, we complement our empirical results from earlier by running all of the experiments using the MT attack. We run MT attack with 20 iterations for each target. Tables B.5 to B.10 provide the results.

We verify that our discussion about accuracy, robustness, and Lipschitzness remains valid using this attack. Comparing with the results using the PGD attack (Tables 3.2 to 3.4), the results with the MT attack gives a slightly lower adversarial test accuracy for all methods. The drop in accuracy is usually around 1–5%. This is within our expectation as this attack is regarded as a stronger attack than PGD.

The MT results still justify the previous discussion from Section 3.5 in general. Training methods leading to models with higher adversarial test accuracy are more locally smooth (smaller local Lipschitz constant during testing). Overall, we believe that seeing consistent results between PGD and MT only strengthens our argument that robustness requires some local Lipschitzness, and moreover, that the accuracy-robustness tradeoff may not be necessary for separated data.

Figure B.4: Separation results for four image datasets. We measure the separation for the original labels, and we also perform the experiment where we randomly label the test and train examples. We see that in MNIST, CIFAR-10, and ResImageNet, the separation diminishes quite a bit when using random labels. Indeed for ResImageNet, there are a number of duplicated examples that appear multiple times in the dataset. Overall, we conclude that the separation is much larger between *different* classes, while this is not the case *within* the same class.

Table B.5: MNIST on CNN1, multi-targeted attack

	train accuracy	test accuracy	adv test accuracy	test lipschitz	gap	adv gap
Natural	100.00	99.20	47.30	67.25	0.80	-0.53
GR	99.99	99.29	89.99	26.05	0.70	3.30
LLR	100.00	99.43	90.49	30.44	0.57	4.06
AT	99.98	99.31	97.23	8.84	0.67	2.65
RST(λ=.5)	100.00	99.34	96.46	11.09	0.66	3.22
RST(λ=1)	100.00	99.31	96.93	11.22	0.69	2.97
RST(λ=2)	100.00	99.31	97.00	12.39	0.69	2.95
TRADES(β=1)	99.81	99.26	96.53	9.69	0.55	2.12
TRADES(β=3)	99.21	98.96	96.60	7.83	0.25	1.34
TRADES(β=6)	97.50	97.54	93.54	2.86	-0.04	0.39

Table B.6: MNIST on CNN2, multi-targeted attack

	train accuracy	test accuracy	adv test accuracy	test lipschitz	gap	adv gap
Natural	100.00	99.51	81.35	23.06	0.49	-0.87
GR	99.99	99.55	92.93	20.26	0.44	2.39
LLR	100.00	99.57	93.76	9.75	0.43	1.70
AT	99.98	99.48	98.01	6.09	0.50	1.94
RST(λ=.5)	100.00	99.53	97.69	8.27	0.47	2.30
RST(λ=1)	100.00	99.55	98.25	6.26	0.45	1.74
RST(λ=2)	100.00	99.56	98.46	4.56	0.44	1.53
TRADES(β=1)	99.96	99.58	98.06	4.74	0.38	1.73
TRADES(β=3)	99.80	99.57	98.54	2.14	0.23	1.18
TRADES(β=6)	99.61	99.59	98.73	1.36	0.02	0.81

Table B.7: SVHN, multi-targeted attack

	train accuracy	test accuracy	adv test accuracy	test lipschitz	gap	adv gap
Natural	100.00	95.85	1.06	149.82	4.15	0.43
GR	96.73	87.80	14.59	40.83	8.94	2.41
LLR	100.00	95.48	20.95	61.64	4.51	3.47
AT	95.20	92.45	49.47	13.03	2.75	14.96
RST(λ=.5)	99.99	93.09	45.98	19.56	6.90	27.26
RST(λ=1)	99.91	93.01	47.06	23.19	6.90	29.19
RST(λ=2)	99.25	92.39	47.58	23.18	6.86	29.99
TRADES(β=1)	98.96	92.45	46.40	18.75	6.51	29.22
TRADES(β=3)	99.33	91.85	49.41	10.15	7.48	32.70
TRADES(β=6)	97.19	91.83	52.82	5.20	5.35	24.28

Table B.8: CIFAR-10, multi-targeted attack

	train accuracy	test accuracy	adv test accuracy	test lipschitz	gap	adv gap
Natural	100.00	93.81	0.00	425.71	6.19	0.00
GR	94.90	80.74	19.15	28.53	14.16	2.88
LLR	100.00	91.44	14.58	94.68	8.56	1.32
AT	99.84	83.51	42.11	26.23	16.33	48.99
RST(λ=.5)	99.90	85.11	38.17	20.61	14.79	32.92
RST(λ=1)	99.86	84.61	39.29	22.92	15.25	38.84
RST(λ=2)	99.73	83.87	40.06	23.95	15.86	41.97
TRADES(β=1)	99.76	84.96	42.22	28.01	14.80	43.69
TRADES(β=3)	99.78	85.55	44.53	22.42	14.23	47.91
TRADES(β=6)	98.93	84.46	46.05	13.05	14.47	43.40

Table B.9: Restricted ImageNet, multi-targeted attack

	train accuracy	test accuracy	adv test accuracy	test lipschitz	gap	adv gap
Natural	97.72	93.47	4.21	32228.51	4.25	-0.24
GR	91.12	88.51	60.61	886.75	2.61	-0.16
LLR	98.76	93.44	50.21	4795.66	5.32	-0.31
AT	96.22	90.33	81.91	287.97	5.90	8.27
RST($\lambda = .5$)	96.78	92.13	78.53	439.77	4.65	4.91
RST($\lambda = 1$)	95.61	92.06	79.33	366.21	3.55	4.68
RST($\lambda = 2$)	96.00	91.14	81.12	390.61	4.86	6.15
TRADES($\beta = 1$)	97.39	92.27	79.46	2144.66	5.13	6.61
TRADES($\beta = 3$)	95.74	90.75	82.00	396.67	5.00	6.35
TRADES($\beta = 6$)	93.34	88.92	81.90	200.90	4.42	5.28

Table B.10: Dropout and generalization. SVHN (perturbation 0.031, dropout rate 0.5) and CIFAR-10 (perturbation 0.031, dropout rate 0.2). We evaluate adversarial accuracy with the multi-targeted attack and compute Lipschitzness with Equation (3.3).

		SVHN					CIFAR-10				
	dropout	test acc.	adv test acc.	test lipschitz	gap	adv gap	test acc.	adv test acc.	test lipschitz	gap	adv gap
Natural	False	95.85	1.06	149.82	4.15	0.87	93.81	0.00	425.71	6.19	0.00
Natural	True	96.66	1.52	152.38	3.34	1.22	93.87	0.00	384.48	6.13	0.00
AT	False	91.68	49.22	16.51	5.11	25.74	83.51	42.11	26.23	16.33	49.94
AT	True	93.05	52.44	11.68	-0.14	6.48	85.20	41.31	31.59	14.51	44.05
RST(λ=2)	False	92.39	47.58	23.17	6.86	36.02	83.87	40.06	23.80	15.86	43.54
RST(λ=2)	True	95.19	50.37	17.59	1.90	11.30	85.49	38.66	34.45	14.00	33.07
TRADES(β=3)	False	91.85	49.41	10.15	7.48	33.33	85.55	44.53	22.42	14.23	47.67
TRADES(β=3)	True	94.00	57.11	4.99	0.48	7.91	86.43	46.38	14.69	12.59	35.03
TRADES(β=6)	False	91.83	52.82	5.20	5.35	23.88	84.46	46.05	13.05	14.47	42.65
TRADES(β=6)	True	93.46	58.53	3.30	0.45	5.97	84.69	49.72	8.13	11.91	26.49

Appendix C

Robustness for Non-parametric Classifiers

C.1 Attack Algorithm: Theoretical Results and Omitted Proofs

In this section, we analyze the exact and approximate region-based attacks. To do so, we provide details about the decompositions for k-NN and tree ensemble classifiers. We also prove Theorem 4.5.3 in general, and we give a corollary for the classifiers that we consider. Finally, we discuss our approximate attack, providing more details and an analysis.

Before getting into these details, we observe that our attack actually holds for the more general class of linear decision trees, which we now define.

Defining Linear Decision Trees

A *linear decision tree* is a binary tree consisting of (i) internal nodes associated with affine functions and (ii) leaf nodes associated with labels in $[C]$. The value $f(\mathbf{x})$ is determined by following the root to a leaf, going left or right depending on whether $\mathbf{x}$ satisfies or violates the linear constraint in the current node; then, $f(\mathbf{x})$ is the label of the leaf. Such trees generalize (standard) decision trees, which restrict each constraint to depend on a single variable.

An *ensemble of linear decision trees* is collection of trees with the modification that the

leaves are labeled with vectors in $\mathbb{R}^C$. The value $f(\mathbf{x})$ is determined by a two-stage process. First, find the root-to-leaf path associated with each tree separately, resulting in a collection of vectors $\mathbf{u}^1, \ldots, \mathbf{u}^T \in \mathbb{R}^C$, where T is the number of trees. Then, letting $\mathbf{u} = \mathbf{u}^1 + \cdots + \mathbf{u}^T$, the output $f(\mathbf{x})$ equals the index of the largest coordinate $i \in [C]$ in the vector $\mathbf{u}$. Note that for binary labels, this is equivalent to the definition of having scalar leaf labels and outputting the sign of the sum.

C.1.1 Decompositions for Specific Classifiers

We now describe the decompositions for tree ensembles and k-NN. Parameters for the decompositions will directly determine the running time of the optimal attack algoggrithm.

Decomposition for Tree Ensembles

Lemma C.1.1. *If f is an ensemble of T linear decision trees, each with depth at most D and with at most L leaves, then f is (L^T, TD)-decomposable.*

Proof. We first describe the decomposition for a single tree, then generalize to an ensemble of trees. Let $\mathcal{T}$ be a linear decision tree with depth D leaves $(\ell_1, \ell_2, \ldots, \ell_m)$. The polyhedron P_i will be the set of $\mathbf{z}$ that reach leaf ℓ_i in $\mathcal{T}$. The hyperplane description for P_i can be computed as follows. Each internal node v from the root of $\mathcal{T}$ to the leaf ℓ_i contains a linear constraint $a_v(\mathbf{z}) \leq b_v$. On the path to ℓ_i, group all the violated (resp. satisfied) constraints a_v, b_v as rows of the matrix A^- and entries of the vector b^- (resp. A^+ and b^+). Then, all $\mathbf{z}$ that reach ℓ_i are exactly the vectors that satisfy $A^-\mathbf{z} > \mathsf{b}^-$ and $A^+\mathbf{z} \leq \mathsf{b}^+$. Therefore, these at most D constraints determine P_i precisely.

Now, consider ensembles of T trees with depth at most D and at most L leaves. The polyhedra correspond to combinations of one leaf from each tree. Each leaf contributes at most D constraints, for at most TD total constraints. There are at most L^T choices for one leaf from each of T trees. □

Decomposition for k-NN

The decomposition for k-NN is a standard fact, known as the k^{th} order Voronoi diagram, and it is a classical result in machine learning and computational geometry (see for example Chapter 12 in the book [145], or the survey [11], or the paper [153]). We sketch a proof for completeness.

Lemma C.1.2. *If f is a k-NN classifier for a dataset of size n, then f is $(\binom{n}{k}, k(n-k))$-decomposable.*

Proof. (Sketch). Let $\mathcal{D}$ be the training dataset on n points. We define $\binom{n}{k}$ convex polyhedra, one for each subset $U \subseteq \mathcal{D}$ containing $|U| = k$ points. The polyhedron P_U is the subset of $\mathbb{R}^d$ such that if $\mathbf{z} \in P_U$, then the k nearest neighbors to $\mathbf{z}$ from the dataset $\mathcal{D}$ in the ℓ_2 distance are the k points in U. By definition, the k-NN classifier will be constant on each polyhedron P_U, as the output label is completely determined by the k nearest neighbors for $\mathbf{z}$, which is the set U.

We show that P_U can be defined by $k(n-k)$ hyperplanes as follows. For each of the k points $\mathbf{x} \in U$, we use the $(n-k)$ bisecting hyperplanes separating $\mathbf{x}$ from each of the $n-k$ points not in U (that is, separating $\mathbf{x}$ from the points $\mathcal{D} \setminus U$). This is a total of $k(n-k)$ linear constraints, and we define P_U as the intersection of these $k(n-k)$ halfspaces. Clearly, P_U is a convex polyhedron.

To see the nearest neighbor property, consider any $\mathbf{z} \in P_U$. For every $\mathbf{x} \in U$, the constraints defining P_U include the $(n-k)$ bisecting hyperplanes that separate $\mathbf{x}$ from the $n-k$ points outside of U. In particular, $\mathbf{z}$ is closer to $\mathbf{x}$ than to these $n-k$ other points. To put this another way, $\mathbf{z}$ is in the Voronoi cell for $\mathbf{x}$ in the reduced dataset consisting only of $\mathbf{x}$ and the other $n-k$ points (that is, $\mathbf{x} \cup (\mathcal{D} \setminus U)$). As this is true for each of the k points in U, we have that $\mathbf{z}$ is closer to each of the k points in U than to the other $n-k$ points. Therefore, we conclude that U consists of the k nearest neighbors to $\mathbf{z}$. □

C.1.2 Analyzing the Region-Based Attack

We have just shown that f is decomposable when it is the classifier determined by k-NN or a linear decision tree (or, more generally, an ensemble of linear decision trees). The consequence of this is that Theorem 4.5.3 implies an efficient and optimal algorithm for a wide-range of non-parametric classifiers. We first discuss the specific convex programs, then finish the proof of the theorem.

Norms as Convex Objectives

Recall that if a classifier is (s,m)-decomposable, then there exists s polyhedra $P_1,\ldots,P_s$ such that each P_i is the intersection of at most m halfspaces. Moreover, the classifier is constant on each of these convex regions, predicting label y_i at all points in P_i.

For an input $\mathbf{x}$, let $I_{\mathbf{x}}$ be the indices of polyhedra P_i such that $f(\mathbf{x}) \neq y_i$. Then, the region-based attack optimizes over all polyhedra P_i for $i \in I_{\mathbf{x}}$ by solving the inner minimization of Equation (4.1), namely

$$\min_{\mathbf{z}\in P_i} \|\mathbf{x}-\mathbf{z}\|_p. \tag{C.1}$$

Given that P_i is a polyhedron, the constraint $\mathbf{z} \in P_i$ can be expressed using the m linear constraints that define P_i. Then, the norm minimization can be expressed as a convex objective. In particular, the problem in Equation (C.1) can be solved with a linear program for $p \in \{1,\infty\}$ or a quadratic program for $p = 2$ using standard techniques [28]. The following are the specific LP formulations for $p \in \{1,\infty\}$.

ℓ_∞ **norm.3** Let $t \in \mathbb{R}$ be single variable. When $p = \infty$, the problem in Equation (C.1) can be solved in $\mathbb{R}^d$ using the following linear program with $d+1$ variables and $m+2d$ linear

constraints.

$$\begin{aligned} \underset{\mathbf{z},t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & \mathbf{z} \in P_i \\ & (\mathbf{z}-\mathbf{x})_j \leq t \qquad \forall\, j \in [d] \\ & (\mathbf{z}-\mathbf{x})_j \geq -t \quad \forall\, j \in [d] \end{aligned} \tag{C.2}$$

ℓ_1 **norm.** Let $\mathbf{t} \in \mathbb{R}^d$ be vector. When $p = 1$, the problem in Equation (C.1) can be solved in $\mathbb{R}^d$ using the following linear program with $2d$ variables and $m + 2d$ linear constraints.

$$\begin{aligned} \underset{\mathbf{z},\mathbf{t}}{\text{minimize}} \quad & \mathbf{1}^T\mathbf{t} \\ \text{subject to} \quad & \mathbf{z} \in P_i \\ & (\mathbf{z}-\mathbf{x})_j \leq \mathbf{t}_j \qquad \forall\, j \in [d] \\ & (\mathbf{z}-\mathbf{x})_j \geq -\mathbf{t}_j \quad \forall\, j \in [d] \end{aligned} \tag{C.3}$$

Finishing the Analysis of the Exact Region-Based Attack

Proof of Theorem 4.5.3. We first claim that the attack produces the optimal adversarial example when f is any (s,m)-decomposable classifier. By assumption, there is a partition of $\mathbb{R}^d$ into polyhedra $P_1, \ldots, P_s$ such that f is constant on each P_i region. Let y_i be the label that f gives to all points in P_i for each $i \in [s]$. On input $\mathbf{x}$, the algorithm considers $i \in I_{\mathbf{x}}$, where $I_{\mathbf{x}} \subseteq [s]$ are the indices such that $f(\mathbf{x}) \neq y_i$. Thus, the point $\mathbf{z}^i \in P_i$ closest to $\mathbf{x}$ will have

$$f(\mathbf{z}^i) = y_i \neq f(\mathbf{x}).$$

Finally, the algorithm's output is

$$\underset{\{\mathbf{z}^i | i \in I_{\mathbf{x}}\}}{\text{argmin}} \ \|\mathbf{z}^i - \mathbf{x}\|.$$

As the regions P_i partition $\mathbb{R}^d$, this is the closest point to $\mathbf{x}$ that receives a different label under f.

We now analyze the running time. For the ℓ_p distance, $p \in \{1, 2, \infty\}$, finding each candidate point $\mathbf{z}^i$ requires solving a convex program with $O(m)$ constraints and $O(d)$ variables. This can be done in $\text{poly}(d, m)$ time using standard optimization techniques (e.g., the interior point method). The number of convex programs is $|I_{\mathbf{x}}| \leq s$. Therefore, the total running time is at most $s \cdot \text{poly}(d, m)$. □

Remark 1 (Targeted Attack)**.** *So far, we have considered untargeted attacks, allowing adversarial examples to have any label other than $f(\mathbf{x})$. An important variation is a* targeted attack*, which specifies a label $\ell \in [C]$, and the goal is to output a close point $\widetilde{\mathbf{x}}$ such that $f(\widetilde{\mathbf{x}}) = \ell$. We note that the region-based attack can be easily modified for this by only searching over $I_{\mathbf{x}}^{\ell} = \{i \in [s] \mid y_i = \ell\}$. This may significantly reduce the running time in practice, as $|I_{\mathbf{x}}^{\ell}|$ may be much smaller than $|I_{\mathbf{x}}|$.*

We specialize the above theorem to ensembles of linear decision trees and the k-NN classifier.

Corollary C.1.3. *Let n be the size of the training set. If $f : \mathbb{R}^d \to [C]$ is a classifier determined by k-NN with $k = O(1)$ or an ensemble of $O(1)$ linear decision trees with depth* $\text{poly}(n)$ *and* $\text{poly}(n)$ *total leaves, then the region-based attack outputs the optimal adversarial example in time* $\text{poly}(d, n)$.

Proof. When f is an ensemble of T linear decision trees, each with depth D and L leaves, Lemma C.1.1 implies that f is (L^T, TD)-decomposable. Assuming that T is a constant and L and D are polynomial means that f is $(\text{poly}(n), \text{poly}(n))$-decomposable. Applying Theorem 4.5.3, the running time of the exact region-based attack is thus $\text{poly}(d, n)$.

When f is the k-NN classifier, Lemma C.1.2 implies that f is $(\binom{n}{k}, k(n-k))$-decomposable. Assuming that k is a constant means that f is $(\text{poly}(n), O(n))$-decomposable. Applying Theorem 4.5.3, the running time of the exact region-based attack is thus $\text{poly}(d, n)$. □

C.2 More Experimental Details

The experiment is run on desktop with Intel Core i7-9700K 3.6 GHz 8-Core Processor and 32 GB of RAM. The code for the experiment is available at `https://github.com/yangarbiter/adversarial-nonparametrics/`.

C.2.1 Classifier Implementation Details

The implementation for DT, RF and k-NN are based on `scikit-learn` [169]. For DT and RF, the splitting criterion is set to "entropy". For computational efficiency, we fixed the maximum depth of DT and RF to be five. For reproducibility, all other hyper-parameters are set to the default parameter settings of the specific implementation.

C.2.2 Attack and Defense Implementation Details

For kernel substitution attack, we set the approximation parameter $c = 1.0$ and attack the substitution model with Projected Gradient Descent (PGD) [144]. For both Region-Based Attacks (RBA-Exact and RBA-Approx), the underlying LP solver that we use is Gurobi [88]. For kernel substitution attack, we use PGD implemented in `Cleverhans` [166]. The implementation of the black-box attack by [50] (BBox) is provided by authors in their public repository[1].

For k-NN, we do not compare with the gradient-based extension [209] attack directly in Section 4.6 since it is under a different setting. Their algorithm only works if k-NN uses the cosine distance instead of ℓ_2 distance.

C.2.3 Dataset Details

For each dataset, we reserve 200 examples for testing. We evaluate the testing accuracy on these 200 examples. To compute empirical robustness $\mathsf{ER}(A, f_D, S, t)$ and defscore(D, A, f, S, t),

[1] `https://github.com/cmhcbb/attackbox`

we randomly select 100 correctly predicted examples for each classifier. For efficiency purposes, the feature dimension for fashion-mnist (f-mnist), mnist is reduced to 25 using principle component analysis (PCA). The original covtype is sub-sampled to 2200 examples. mnist17 represents a subset of mnist dataset for the binary classification problem distinguishing between 1 and 7. Similarly, f-mnist35 is the task of distinguishing between 3rd and 5th class, and f-mnist06 is the task of distinguishing between 0th and 6th class. The features are scaled to $[0,1]$ so the solver will avoid numerical rounding errors.

Table C.1: Dataset statistics.

	# train	# test (perturb.)	# test (accuracy)	features	classes
austr.	490	100	200	14	2
cancer	483	100	200	10	2
covtype	2000	100	200	54	2
diabetes	568	100	200	8	2
f-mnist35	12000	100	200	25	2
f-mnist06	12000	100	200	25	2
fourclass	662	100	200	2	2
halfmoon	2000	100	200	2	2
mnist17	13007	100	200	25	2

C.2.4 Additional Experiment Results

Tables C.2 to C.5 show additional experiment results with adversarial pruning (AP) as defense. In these tables, for AP with separation parameter $r = 0.5$, we have some invalid values. These values are caused by setting a too large value of r which results int that the adversarial pruned datasets to be highly unbalanced in label or even making the dataset have a single label left. If the training accuracy goes below 0.5 or the prediction of the classifier outputs only one label, we will put the value being "-" in the table. For diabetes with 3-NN, its caused by 3-NN only predicts one label.

Testing accuracy is a sanity check that we are not giving away all accuracy for robustness.

The higher the empirical robustness is means the classifier is more robust to the given attack. When considering the strength of the attack, empirical robustness is lower the better. When considering the strength of the defense, defscore is higher the better. For defscore higher mean that after defense (AP), the classifier become more robust, thus higher the better.

Table C.2: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for 1-NN.

	1-NN			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.151	.805	490	.162	.800	484	1.073	.249	.820	458	1.649	.311	.825	427	2.060
cancer	.137	.950	483	.137	.950	483	1.000	.193	.950	473	1.409	.261	.965	458	1.905
covtype	.066	.725	2000	.072	.700	1904	1.091	.289	.685	1417	4.379	.346	.675	1384	5.242
diabetes	.035	.695	568	.035	.700	535	1.000	.164	.660	379	4.686	.375	.660	370	10.714
f-mnist06	.029	.800	12000	.031	.820	11509	1.069	.075	.765	7348	2.586	-	.495	6000	-
f-mnist35	.075	1.000	12000	.075	1.000	11999	1.000	.089	.980	10477	1.187	.104	.945	8139	1.387
fourclass	.090	1.000	662	.107	.960	559	1.189	.278	.750	453	3.089	-	.565	442	-
halfmoon	.058	.920	2000	.151	.915	1702	2.603	.161	.840	1144	2.776	-	.480	1004	-
mnist17	.070	.975	13007	.072	.975	13004	1.029	.097	.965	11128	1.386	.118	.810	6783	1.686

Table C.3: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for 3-NN.

	3-NN			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.278	.805	490	.317	.810	484	1.140	.333	.815	458	1.198	.371	.825	427	1.335
cancer	.204	.975	483	.204	.975	483	1.000	.283	.960	473	1.387	.350	.970	458	1.716
covtype	.108	.750	2000	.117	.735	1904	1.083	.357	.685	1417	3.306	.394	.680	1384	3.648
diabetes	.078	.755	568	.078	.750	535	1.000	.232	.655	379	2.974	-	.660	370	-
f-mnist06	.051	.795	12000	.050	.825	11509	.980	.089	.750	7348	1.745	-	.495	6000	-
f-mnist35	.094	1.000	12000	.093	1.000	11999	.989	.108	.985	10477	1.149	.121	.950	8139	1.287
fourclass	.096	.995	662	.127	.960	559	1.323	.297	.750	453	3.094	-	.565	442	-
halfmoon	.096	.940	2000	.159	.920	1702	1.656	.184	.845	1144	1.917	-	.480	1004	-
mnist17	.097	.985	13007	.094	.985	13004	.969	.110	.960	11128	1.134	.141	.795	6783	1.454

Defense figures

Figures C.1 and C.2 show the complete experiment results for the experiment in Figure 4.3. The accuracy (y-axis) is measured on the 100 correctly predicted testing examples sampled initially.

Table C.4: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for DT.

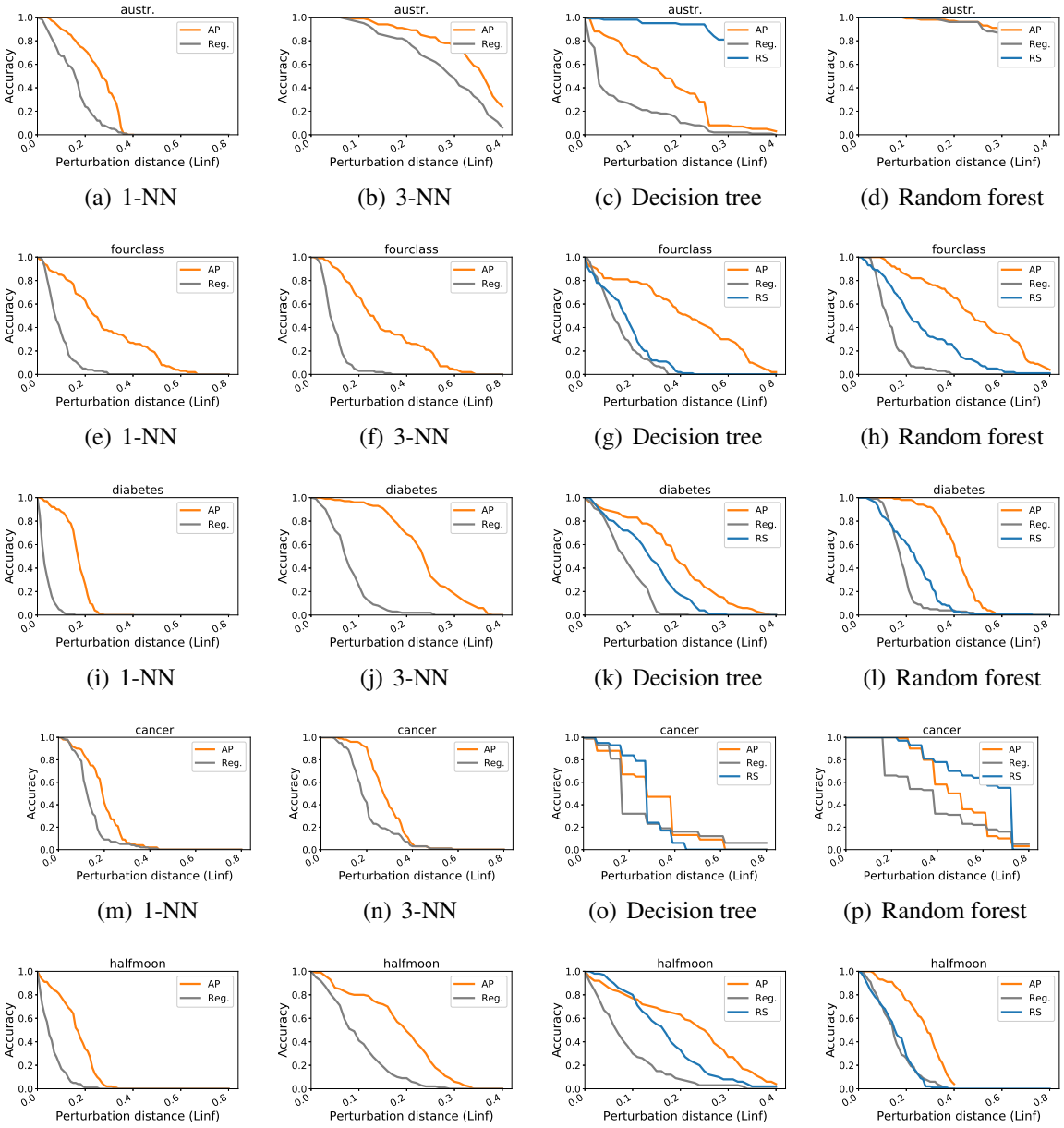
	DT			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.070	.855	490	.194	.835	484	2.771	.166	.835	458	2.371	.450	.835	427	6.429
cancer	.255	.930	483	.255	.930	483	1.000	.303	.965	473	1.188	.358	.960	458	1.404
covtype	.051	.715	2000	.051	.740	1904	1.000	.230	.680	1417	4.510	.221	.665	1384	4.333
diabetes	.085	.715	568	.085	.720	535	1.000	.189	.670	379	2.224	.378	.670	370	4.447
f-mnist06	.079	.805	12000	.092	.825	11509	1.165	.203	.770	7348	2.570	-	.495	6000	-
f-mnist35	.115	.995	12000	.110	.995	11999	.957	.237	.940	10477	2.061	.281	.925	8139	2.443
fourclass	.137	.900	662	.138	.910	559	1.007	.416	.680	453	3.036	-	.565	442	-
halfmoon	.085	.950	2000	.167	.895	1702	1.965	.219	.670	1144	2.576	-	.480	1004	-
mnist17	.123	.975	13007	.126	.970	13004	1.024	.162	.955	11128	1.317	.316	.830	6783	2.569

Table C.5: The number of training data left after adversarial pruning (AP), testing accuracy, empirical robustness, and defscore with different separation parameter of AP for RF.

	RF			AP (separation parameter r=.1)				AP (separation parameter r=.3)				AP (separation parameter r=.5)			
	ER	test accuracy	# train	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore	ER	test accuracy	# train	defscore
austr.	.446	.845	490	.426	.855	484	.955	.465	.840	458	1.043	.496	.835	427	1.112
cancer	.383	.970	483	.383	.970	483	1.000	.481	.965	473	1.256	.496	.955	458	1.295
covtype	.214	.750	2000	.226	.700	1904	1.056	.456	.680	1417	2.131	.481	.695	1384	2.248
diabetes	.184	.755	568	.175	.740	535	.951	.409	.660	379	2.223	.710	.660	370	3.859
f-mnist06	.188	.790	12000	.215	.785	11509	1.144	.333	.755	7348	1.771	-	.495	6000	-
f-mnist35	.246	1.000	12000	.236	.995	11999	.959	.346	.925	10477	1.407	.289	.925	8139	1.175
fourclass	.133	.980	662	.181	.865	559	1.361	.478	.665	453	3.594	-	.565	442	-
halfmoon	.149	.930	2000	.198	.900	1702	1.329	.271	.755	1144	1.819	-	.480	1004	-
mnist17	.250	.970	13007	.230	.965	13004	.920	.314	.945	11128	1.256	.359	.800	6783	1.436

C.2.5 Images Removed by Adversarial Pruning

Figure C.3 shows examples of images removed by AP and their closest opposite labeled images. In the case of mnist17, it is interesting to note that the sevens and ones do resemble each other, and so it makes sense that they are close to the training boundary and should be pruned. For the other datasets, recall that we first applied PCA, and therefore, these images are similar in the resulting feature space.



(a) 1-NN (b) 3-NN (c) Decision tree (d) Random forest

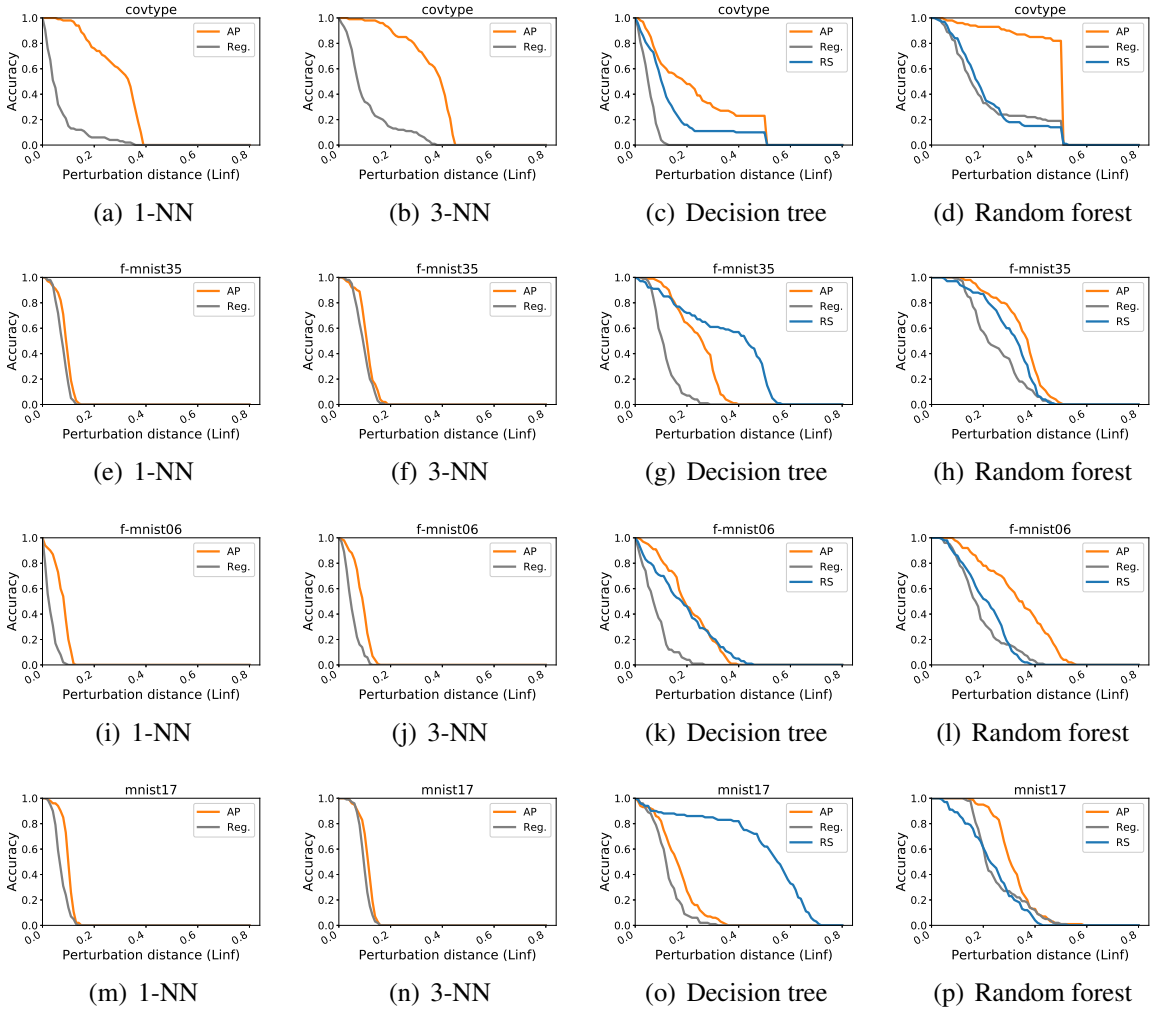
(e) 1-NN (f) 3-NN (g) Decision tree (h) Random forest

(i) 1-NN (j) 3-NN (k) Decision tree (l) Random forest

(m) 1-NN (n) 3-NN (o) Decision tree (p) Random forest

(q) 1-NN (r) 3-NN (s) Decision tree (t) Random forest

Figure C.1: The maximum perturbation distance allowed versus the accuracy on the 100 correctly predicted test examples (see Appendix C.2.3 for details).



(a) 1-NN (b) 3-NN (c) Decision tree (d) Random forest

(e) 1-NN (f) 3-NN (g) Decision tree (h) Random forest

(i) 1-NN (j) 3-NN (k) Decision tree (l) Random forest

(m) 1-NN (n) 3-NN (o) Decision tree (p) Random forest

Figure C.2: The maximum perturbation distance allowed versus the accuracy on the 100 correctly predicted test examples (see Appendix C.2.3 for details).

(a) mnist17 (b) mnist17 (c) mnist17 (d) mnist17

(e) f-mnist35 (f) f-mnist35 (g) f-mnist35 (h) f-mnist35

(i) f-mnist06 (j) f-mnist06 (k) f-mnist06 (l) f-mnist06

Figure C.3: Examples of images removed by adversarial pruning (AP). The images removed are (a), (c), (e), (g), (i), (k) and the images to its right are the closest image with opposite labeled. To interpret the labels of these datasets, mnist17 is the task of classifying one versus seven, f-mnist35 is Dress versus Sandal, and f-mnist06 is T-Shirt/top versus Shirt.

Appendix D

Connecting Interpretability and Robustness in Decision Trees through Separation

D.1 Proofs

D.1.1 Separation and Interpretability

Theorem 5.4.1. *For any tree size s and $\gamma > 0$, there is a dataset in $\mathbb{R}^2$ that is linearly separable, and any decision tree with size s has accuracy less than $\frac{1}{2} + \gamma$.*

Proof. Fix an integer s and $\gamma > 0$. We will start by describing the dataset and prove it is linearly separable. We will then show that any decision tree of size s must have accuracy smaller than $1/2 + \gamma$. **Dataset.** The dataset size is n, to be fixed later. The dataset includes, for any $i = 1 \dots n/4$, a group, G_i, of four points: two points $(i, -i + \varepsilon), (i + \varepsilon, -i)$ that are labeled positive and two points $(i, -i - \varepsilon), (i - \varepsilon, -i)$ that are labeled negative for some small $\varepsilon > 0$, see Figure D.1.

To prove that the dataset is linearly separable focus on the vector $w = (0.5, 0.5)$ with $|w|_1 = 1$. For each labeled examples (x, y) in the dataset, the inner product is equal to $yw \cdot x = \varepsilon/2 > 0$.

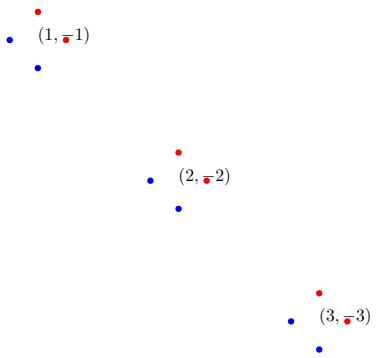


Figure D.1: Proof of Theorem 5.4.1. Linearly separable dataset that is not interpretable by a small decision tree. Around point $(i,-i)$ there are 4 close points: two points $(i,-i+\varepsilon),(i+\varepsilon,-i)$ that are labeled positive and two points $(i,-i-\varepsilon),(i-\varepsilon,-i)$ that are labeled negative for some small $\varepsilon > 0$.

Accuracy.

We will prove by induction that the points arriving in each node are a series of consecutive groups, perhaps except a few points that are in the "tails". In each node, the number of positive and negative examples is about the same, so one cannot predict well. See intuition in Figure D.2.

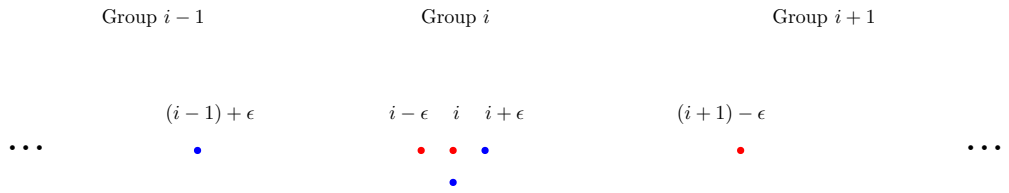


Figure D.2: Projection of the points in the dataset to the first feature. We see that the groups appear one after another, each cut defined by an inner node in the tree, will leave the order between the groups as is. In a series of consecutive groups, the number of positive and negative examples is equal. Thus, the accuracy is close to $1/2$. Similar figure when projecting to the second feature.

More formally, a tail G'_i is a subset of G_i, i.e., $G'_i \in \wp(G_i)$, where $\wp$ is the power set. We prove the following claim.

To finish the proof of Theorem 5.4.1, first note that the number of positive and negative examples in each G_i is exactly equal. Together with the claim that we just proved, for each leaf v, the number of points that are correctly classified out of the points, P_v, reaching v, is at most $|P_v|/2+4$. Thus the total number of points correctly classified by the entire tree is at most

$n/2+4s$. Thus, the accuracy of the tree is at most $1/2+4s/n$. We take $n > 4s/\gamma$ and get that the accuracy is smaller than $1/2+\gamma$, which is what we wanted to prove.

□

Theorem 5.4.2. *For any labeled data in* $[-1,1]^d \times \{-1,1\}$ *that is r-separated, there is a decision tree of depth at most* $\frac{6d}{r}$ *which has a training error* 0.

Proof. Each feature is in $[-1,1]$, so $r \leq 1$. Fix $r \leq \Delta < 2r$ and $L = \lceil \frac{1}{\Delta} \rceil$. We can bound L by

$$L \leq \frac{1}{\Delta} + 1 \leq \frac{1}{r} + 1 \leq \frac{2}{r}.$$

Take two data points, one labeled positive, x^+, and one negative x^-. By the r-separation, we know that there is a feature i' such that

$$|x_{i'}^+ - x_{i'}^-| \geq 2r > \Delta.$$

This means that we can find a threshold θ among the $2L+1$ thresholds $-L\cdot\Delta, \ldots, 0\cdot\Delta, 1\cdot\Delta, \ldots, L\cdot\Delta$ that distinguishes the examples x^+ and x^-, i.e., there is $j' \in \{-L, \ldots, L\}$, such that

$$sign(x_{i'}^+ - \Delta \cdot j') \neq sign(x_{i'}^- - \Delta \cdot j').$$

We focus on the decision tree with all possible features and the $2L+1$ thresholds. All examples reaching the same leaf, has the same label. In other words, the training error is 0. Since there are $d \cdot (2L+1)$ pairs of feature and thresholds, the depth of the tree is at most $d(2L+1) \leq 3dL \leq \frac{6d}{r}$.

□

Theorem 5.4.3. *There is a labeled dataset in* $[-1,1]^d$ *which is* 1*-separated and has the following property. For any* $\gamma > 0$ *and any decision tree* T *that achieves accuracy* $0.5+\gamma$*, the size of* T *is at least* $\gamma 2^d$.

Proof. We start with describing the dataset, then we will show that any decision tree must be large if we want to achieve $0.5+\gamma$ accuracy.

The dataset. The inputs are all strings in $\{-1,1\}^d$. The hypothesis is the parity function, i.e., and

$$f(x_1,\ldots,x_d) = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} \frac{x_i+1}{2} \equiv 1 \pmod 2 \\ -1 & \text{o.w.} \end{cases}$$

Each node has equal number of positive and negative examples. The main idea is that as long as the depth of a node is not d, it has exactly the same number of positive and negative examples reaching it. This is true since as long as at most $d-1$ features are fixed, there exactly the same number of positive and negative examples that agree on these features.

Large size. Denote by N_1 the set of all nodes that exactly one example reach them. Denote this set size by $|N_1| = n_1$. We want to prove that n_1 is large. So far, we proved that for each node that contains more than one example, exactly half of the examples are labeled positive and half negative. Number of examples correctly classified is half of all the examples not in N_1 plus n_1. There are 2^d examples in total. So the accuracy is equal to $\frac{(2^d-n_1)/2+n_1}{2^d} = \frac{1}{2} + \frac{n_1}{2^{d+1}}$. The latter should be at least $1/2+\gamma$. Therefore, the size of the tree is at least $\gamma 2^d$.

□

D.1.2 Linear Separability: Weak Learner

We start with the more restricted case where the features are binary. This will give the necessary foundations for the general case, where features are in $[-1,1]$. In this case $\mathcal{H}_t$ is simplified to the set $\{x \mapsto x_i : i \in [d]\}$.

Theorem D.1.1. *For any data in $\{-1,1\}^d \times \{-1,1\}$ that is labeled by a γ-linearly separable hypothesis f and for any distribution μ on the examples, there is hypothesis $h \in \mathcal{H}_t$ such that*

$$\Pr_{x\sim\mu}(h(x) = f(x)) \geq \frac{1}{2} + \frac{\gamma}{2}.$$

Proof. The proof's high-level idea is to represent the class as a bipartite graph and lower bound

the weighted density of this graph. The high lower-bound leads to a high degree vertex, which will correspond to our desired weak learner.

Recall that by the fact that the data is γ-linearly separable, we know that there is a vector w with $|w|_1 = 1$ such that for eac labeled example (x, y) it holds that

$$yw \cdot x \geq \gamma. \tag{D.1}$$

Bipartite graph description. Consider the following bipartite graph. The vertices are the m examples in the training data and the d hypotheses in the binary version of $\mathcal{H}_t$. There is an edge $(x, h_i) \in E$ between an example x and hypothesis h_i, if it correctly classify x, i.e., if $f(x) = h_i(x)$. Each vertex is given a weight: example x^j gets weight μ_j and a hypothesis h_i gets weight w_i, where w is in Equation (D.1).

Assumption: $\sum_{i=1}^{d} w_i = 1$. We assume without loss of generality that $w_i \geq 0$, otherwise if there is $w_i < 0$, we can multiply the i-th feature in all the examples by -1. After this multiplication, the linearity assumption still holds. We know that $\sum_{i=1}^{d} |w_i| = 1$, and since $w_i \geq 0$, we get that $\sum_{i=1}^{d} w_i = 1$.

Lower bound weighted-density. The main idea is to lower bound the weighted density ρ of the bipartite graph, which is the sum over all edges (h_i, x^j) in the graph, each with weight w_i, μ_j:

$$\rho = \sum_{(x^j, h_i) \in E} w_i \mu_j.$$

To prove a lower bound on ρ, we focus on one labeled example (x^j, y^j). From the linearity assumption, Equation (D.1), we know that $\sum_{i=1}^{d} y^j w_i x_i^j \geq \gamma$. Recall that $y^j x_i^j$ is equal to $+1$ or -1, since we are in the binary case. We can separate the sum depending on whether $y^j x_i^j$ is equal to $+1$ or -1 and get that

$$\sum_{i: y^j x_i^j = 1} w_i - \sum_{i: y^j x_i^j = -1} w_i \geq \gamma.$$

We know that $\sum_{i=1}^{d} w_i = 1$, thus $\sum_{i:y^j x_i^j = -1} w_i = 1 - \sum_{i:y^j x_i^j = 1} w_i$, and the inequality can be rewritten as

$$2 \sum_{i:y^j x_i^j = 1} w_i - 1 \geq \gamma.$$

Notice that $(x^j, h_i) \in E \Leftrightarrow y^j x_i^j = 1$. Thus, the inequality can be further rewritten as

$$\sum_{i:(x^j,h_i)\in E} w_i \geq \frac{1+\gamma}{2}.$$

This inequality holds for any labeled example, so we can sum all these inequalities, each with weight μ_j and get that

$$\rho \geq (1+\gamma)/2.$$

Finding a weak learner. Since w is a probability distribution, we can rewrite ρ and get

$$\mathbb{E}_{i\sim w}\left[\sum_{j:(x^j,h_i)\in E} \mu_j\right] \geq (1+\gamma)/2.$$

From the probabilistic method [4], there is a hypothesis h_i such that

$$\sum_{j:(x^j,h_i)\in E} \mu_j \geq (1+\gamma)/2.$$

By the definition of the graph, $(x^j, h_i) \in E \Leftrightarrow h_i(x^j) = f(x^j)$. Thus, we get that

$$\Pr_{x\sim\mu}(h_i(x) = f(x)) \geq \frac{1}{2} + \frac{\gamma}{2},$$

which is exactly what we wanted to prove.

□

Theorem D.1.2 (binary weak-learner)**.** *For any distribution μ over labeled examples $\{-1,+1\}^d \times$*

$\{-1,+1\}$ *that satisfies linear separability with a* γ*-margin, and for any* $\delta \in (0,1)$ *there is* $m = O\left(\frac{d+\log\frac{1}{\delta}}{\gamma^2}\right)$*, such that with probability at least* $1-\delta$ *over the sample S of size m, it holds that*

$$\Pr_{(x,y)\sim\mu}(h_S(x) = y) \geq \frac{1}{2} + \frac{\gamma}{4}.$$

Proof. We start with a known fact[1] that

$$VC(\mathcal{H}_t) \leq d.$$

Denote the best hypothesis in the binary version in $\mathcal{H}_t$ as h^*. From Theorem D.1.1, we know that $\Pr_x(h^*(x) = f(x)) \geq \frac{1}{2} + \frac{\gamma}{2}$. For every sample S, denote by h_S the hypothesis in the binary version of $\mathcal{H}_t$ that optimizes the accuracy on the sample S. From the fundamental theorem of statistical learning [202], we know that for $m = O\left(\frac{d+\log\frac{1}{\delta}}{(\gamma/4)^2}\right)$, with probability at least $1-\delta$ over the sample S of size m, it holds that

$$\Pr_{(x,y)\sim\mu}(h_S(x) = y) \geq \Pr_{(x,y)\sim\mu}(h^*(x) = y) - \frac{\gamma}{4} \geq \frac{1}{2} + \frac{\gamma}{4}.$$

□

Theorem 5.5.1. *Fix* $\alpha > 0$*. For any data in* $[-1,1]^d \times \{-1,1\}$ *that is labeled by a* γ*-linearly separable hypothesis* f *and for any distribution* μ *on the examples, there is a hypothesis* $h \in \mathcal{H}_t$ *such that*

$$\Pr_{\mathbf{x}\sim\mu}(h(\mathbf{x}) = f(\mathbf{x})) \geq \frac{1}{2} + \frac{\gamma}{2} - \alpha$$

.

Proof. The proof is similar in spirit to the proof of Theorem D.1.1. The main difference is the technique to lower bound ρ, the weighted density. In the current proof we use the fact that if for

[1] To bound the $VC(\mathcal{H}_t)$ note that for any $d+1$ points in $\mathbb{R}^d$ there is at least one point v, where in each coordinate it is not the largest one among the $d+1$ points. Thus, it's impossible that v is labeled $+1$ while all the rest of the points are labeled -1.

some positive example x, its i-th feature, x_i, is high, then it contains many edges in the graph. For a negative example x, if its i-th feature, x_i, is low, then it contains many edges in the graph.

Bipartite graph description. We first discretize the segment $[-1,1]$ to $\ell \geq 2/\alpha + 1$ values with $\Delta = 2/(\ell - 1)$:

$$Z = \{-1, -1+\Delta, \ldots, 1-\Delta, 1\}.$$

The value of Δ was chosen such that $|Z| = \ell$. We focus on the following subclass of $\mathcal{H}_t'$ which is a discretization of $\mathcal{H}_t$

$$\mathcal{H}_t' = \{h_{i,\theta}\} \quad \text{for } i \in [d], \theta \in Z.$$

We use a similar bipartite graph as in the proof of Theorem D.1.1 for the subclass $\mathcal{H}_t'$: the vertices are the m examples and the hypotheses in $\mathcal{H}_t'$; and there is an edge $(x, h_{i,\theta}) \in E$ between an example x with label y and hypothesis $h_{i,\theta}$ whenever $yx_i \geq \theta$, i.e., when $h_{i,\theta}$ correctly classify x.

Recall that by the fact that the data is γ-linearly separable, we know that there is a vector w with $|w|_1 = 1$ and for any labeled example (x,y) it holds that

(D.2)

From the same argument as in Theorem D.1.1, we assume that $\sum_{i=1}^{d} w_i = 1$.

We are now ready to give each vertex in the bipartite graph a weight: example x^j gets weight μ_j and a hypothesis $h_{i,\theta}$ gets weight $w_{i,\theta} = w_i/\ell$, where w is as in Equation (D.2). The weights on the hypotheses were chosen such that they will sum up to 1:

$$\sum_{i,\theta} w_{i,\theta} = \sum_{i,\theta} \frac{w_i}{\ell} = \sum_i w_i = 1. \tag{D.3}$$

Lower bound weighted-density. We will lower bound the weighted density ρ of the

bipartite graph, which is the sum over all edges $(x^j, h_{i,\theta})$ in the graph, each with weight $w_{i,\theta}\mu_j$:

$$\rho = \sum_{(x^j, h_{i,\theta}) \in E} w_{i,\theta}\mu_j.$$

To show a lower bound on ρ, we focus on one labeled example (x, y) and one feature i and we will lower bound the following sum

$$\sum_{\theta:(x,h_{i,\theta})\in E} w_{i,\theta} = \frac{w_i}{\ell} \sum_{\theta:(x,h_{i,\theta})\in E} 1. \tag{D.4}$$

The sum in the RHS is equal to the number of θ's such that $yx_i^j \geq \theta$:

(D.5)

To lower bound $d_{x,i}$ we separate the analysis depending on the label y. If x is a positive example, i.e., $y = +1$, we have that

$$d_{x,i} \geq (1 + x_i - \Delta)/\Delta$$

For a negative example x, we can lower bound $d_{x,i}$ by

$$d_{x,i} \geq (1 - x_i - \Delta)/\Delta$$

To summarize these two equations we get that

$$d_{x,i} \geq (1 + yx_i - \Delta)/\Delta$$

Going back to Equation (D.4),

$$\sum_{\theta:(x,h_{i,\theta})\in E} w_{i,\theta} \geq \frac{w_i}{\ell} \cdot \frac{1+yx_i-\Delta}{\Delta} = w_i(1+yx_i-\Delta) \cdot \frac{\ell-1}{2\ell}.$$

Summing over all features

$$\begin{aligned} \sum_{i,\theta:(x,h_{i,\theta})\in E} w_{i,\theta} &= \frac{\ell-1}{2\ell}\left(\sum_i w_i(1-\Delta) + \sum_i yx_i w_i\right) && \text{(D.6)} \\ &\geq \frac{\ell-1}{2\ell}(1+\gamma-\Delta) = \left(\frac{1}{2}+\frac{\gamma}{2}-\frac{\Delta}{2}\right)\left(1-\frac{1}{\ell}\right) && \text{(D.7)} \end{aligned}$$

Weighted sum over all examples yields the desired lower bound on ρ

$$\rho \geq \left(\frac{1}{2}+\frac{\gamma}{2}-\frac{\Delta}{2}\right)\left(1-\frac{1}{\ell}\right) \geq \frac{1}{2}+\frac{\gamma}{2}-\alpha.$$

Finding a weak learner. The proof now continues similarly to the proof of Theorem D.1.1. Since $w_{i,\theta}$ is a probability distribution over all hypotheses in $\mathcal{H}_t'$ (see Equation (D.3)), we can rewrite ρ and get

$$\mathbb{E}_{h_{i,\theta}\sim w}\left[\sum_{j:(x^j,h_{i,\theta})\in E} \mu_j\right] \geq \frac{1}{2}+\frac{\gamma}{2}-\alpha.$$

From the probabilistic method [4], there is hypothesis h_i such that

$$\sum_{j:(x^j,h_i)\in E} \mu_j \geq \frac{1}{2}+\frac{\gamma}{2}-\alpha.$$

By the definition of the graph, $(x^j,h_i)\in E \Leftrightarrow h_i(x^j)=f(x^j)$. Thus, we get that

$$\Pr_{x\sim\mu}(h_i(x)=f(x)) \geq \frac{1}{2}+\frac{\gamma}{2}-\alpha,$$

which is exactly what we wanted to prove.

□

Theorem 5.5.2 (weak-learner)**.** *Fix $\alpha > 0$. For any distribution μ over $[-1,+1]^d \times \{-1,+1\}$ that satisfies linear separability with a γ-margin, and for any $\delta \in (0,1)$ there is $m = O\left(\frac{d+\log\frac{1}{\delta}}{\gamma^2}\right)$, such that with probability at least $1-\delta$ over the sample S of size m, it holds that*

$$\Pr_{(\mathbf{x},y)\sim\mu}(h_S(\mathbf{x}) = y) \geq \frac{1}{2} + \frac{\gamma}{4} - \alpha.$$

Proof. Proof is similar to Theorem D.1.2. □

D.1.3 Linear Separability: Risk Scores

Claim 5.5.1. *If every condition in a risk-score model R is of the form "$x_i \geq \theta$" and all weights are positive, except the bias term, then R is a monotone model.*

Proof. Fix a risk score model f which is defined by a series of m conditions "$x_{i_1} \geq \theta_1$", ..., "$x_{i_m} \geq \theta_m$" and weights $w_0, w_1, \ldots, w_m$. The score, $s(z)$, of an example z is the weighted-sum of all satisfied conditions,

$$s(z) = w_0 + \sum_{j=1}^{m} w_j I_{z_j \geq \theta_j},$$

where $I_A = 1$ if A is true, and $I_A = 0$ otherwise. The prediction of the model f is equal to

$$f(z) = sign(s(z)).$$

Fix examples $x, y \in \mathbb{R}^d$ with $x \leq y$. Our goal is to show that $f(x) \leq f(y)$. The key observation is that any condition $x_{i_j} \geq \theta_j$ satisfied by x is also satisfied by y because $y_{i_j} \geq x_{i_j} \geq \theta_j$, by our assumption that $x \leq y$. In different words we have that

(D.8)

This implies that the score of y is at least the score of x, since it holds that

$$s(y) = w_0 \sum_{j=1}^{m} w_j I_{y_j \geq \theta_j} \geq w_0 + \sum_{j=1}^{m} w_j I_{x_j \geq \theta_j} = s(x).$$

Thus, we get exactly what we wanted to prove $f(y) = sign(s(y)) \geq sign(s(x)) = f(x)$.

As an aside, at this point, it should become apparent why we restricted our conditions to be of the form "$x_{i_j} \geq \theta_j$" and did not allow natural conditions of the form "$x_{i_j} \leq \theta_j$", such conditions will not be monotone and Inequality (Equation (D.8)) will not hold. □

Claim 5.5.2. *Assume a learning algorithm A gets as an input a sample from a γ-linearly separable data and returns a* monotone *model with accuracy $1-\varepsilon(\gamma)$. Then, there is an algorithm that returns a model with astuteness (Definition 2) at least $1-\varepsilon\left(\frac{\gamma}{2}\right)$ at radius $\gamma/2$.*

Proof. The high-level idea is to focus on a noisier distribution than the original one. The noise is small enough so that the new data will remain linearly separable with a (slightly worse) margin. Therefore, the learning algorithm can be applied. We call the learning algorithm with a sample from the noisy distribution and return its result. We will prove that a point correctly classified in the noisy dataset implies that the noiseless point is robust to adversarial examples.

Noisy data. Fix a data $D \subseteq \mathcal{X} \times \{-1,+1\}$ and a distribution μ on D. We will create a noisy distribution μ' on the labeled examples by mapping each example $(x,y) \in D$ to a new labeled example (x',y) where

$$x' = x - y\frac{\gamma}{2}\mathbf{1},$$

where $\mathbf{1}$ is the vector of all 1. Thus, if x is a positive labeled example ($y=+1$) then $x' = x - \frac{\gamma}{2}\mathbf{1}$. This means that from each coordinate we decrease $\gamma/2$. Intuitively, we make x looks more negative. Similarly, if x is a negative labeled example ($y=-1$) we create a new example $x' = x + \frac{\gamma}{2}\mathbf{1}$, intuitively, making x looks more positive by adding $\gamma/2$ to each coordinate.

If we have a samples from μ, then we can easily sample from μ' by subtracting $y\frac{\gamma}{2}\mathbf{1}$ from each labeled example (x,y) in the sample.

Noisy data is $\gamma/2$-linearly separable. Suppose that the original data D is γ-separable. This means that there is a vector w with $|w|_1 = 1$ such that for every labeled example $(x,y) \in D$ in it holds that $yw \cdot x \geq \gamma$. We know that the corresponding example in the noisy data is equal to $x' = x - y\frac{\gamma}{2}\mathbf{1}$. We will prove a lower bound on $yw \cdot x'$ and this will prove that the noisy input is also linearly separable with a margin. We will use the fact that $y^2 = 1$ for any label y and get that

$$yw \cdot x' = yw\left(x - y\frac{\gamma}{2}\mathbf{1}\right) = ywx - y^2 w \cdot \frac{\gamma}{2}\mathbf{1} = ywx - w \cdot \frac{\gamma}{2}\mathbf{1}$$

Recall that $|w|_1 = 1$, which means that $w \cdot \mathbf{1} = \sum_i w_i \leq \sum_i |w_i| = |w|_1 = 1$. This means that $-w \cdot \frac{\gamma}{2}\mathbf{1} \geq -\gamma/2$. Together with the fact that $yw \cdot x \geq \gamma$, we can now give the desired lower bounds on $yw \cdot x'$

$$yw \cdot x' \geq ywx - \gamma/2 \geq \gamma - \gamma/2 = \gamma/2.$$

In different words, the new data is $\gamma/2$-linearly separable.

Model is robust. In order to construct a robust model, we take our sample S from μ. Then we transform it to a sample from μ' by subtracting noise of $y\frac{\gamma}{2}\mathbf{1}$ to each labeled example (x,y), and then we call algorithm A with the noisy training data. From our assumption, the resulting model has accuracy $1 - \varepsilon(\frac{\gamma}{2})$. We will show that the returned model has astuteness of $1 - \varepsilon(\frac{\gamma}{2})$ at radius $\gamma/2$ with respect to μ. We do this by proving that if (x',y) is correctly classified than the model is robust at x with radius $\gamma/2$.

Fix a noisy labeled example (x',y) and an example z that is $\gamma/2$ close to x, in ℓ_∞, i.e., $\|x - z\|_\infty \leq \gamma/2$.

□

D.2 Additional Experiment Details

D.2.1 Setups

The experiments are performed on a Intel Core i9 9940X machine with 128GB of RAM. The code for the experiments is available at this url[2].

Additional dataset details. The adult, bank, breastcancer, mammo, mushroom, and spambase datasets are retrieved from a publicly available repository[3], and these datasets are used by Ustun and Rudin [224]. The careval, ficobin, and campasbin datasets are also retrieved from a publicly available source[4] and used by Lin et al. [136]. We also added the diabetes, heart, and ionosphere dataset from[5] and bank2 dataset from Moro et al. [150]. All features are scaled to $[0,1]$ by the following formula $(x-min)/(max-min)$, where x represents the feature value, and *min* and *max* represents the minimum and maximum value of the given feature across the entire data.

In the adult dataset, the target is to predict whether the person's income is greater then $50,000$. For bank and bank2 datasets, we want to predict whether the client opens a bank account after a marketing call. In the breastcancer dataset, we want to predict whether the given sample is benign. In the heart dataset, we want to detect the presence of heart disease in a patient. In the mammo dataset, we want to predict whether the sample from the mammography is malignant. In the mushroom dataset, we want to predict whether the mushroom is poisonous. In the spambase dataset, we want to predict whether an email is a spam. In the careval dataset, the goal is to evaluate cars. In the ficobin dataset, we want to predict a person credit risk. In the campasbin dataset, we want to predict whether a convicted criminal will re-offend again. In the diabetes dataset, we want to predict whether or not the patients in the dataset have diabetes. In the ionosphere dataset, we want to predict whether the radar data are showing some evidence of some type of structure in the ionosphere.

[2] https://github.com/yangarbiter/interpretable-robust-trees

[3] https://github.com/ustunb/risk-slim/tree/master/examples/data

[4] https://github.com/Jimmy-Lin/GeneralizedOptimalSparseDecisionTrees/tree/master/experiments/datasets

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Baseline implementations. For DT, we use the implementation from `scikit-learn` [169] and set the splitting criteria to entropy. For LCPA and RobDT, we use the implementation from their original authors[6].

Measure empirical robustness. The IR for DT and RobDT can be measured using the method in [108, 239]. The IR for BBM-RS is measured using the method in [6]. The IR for LCPA can be measured by solving a linear program.

D.3 Additional Results

D.3.1 Examples That Are Similar but Labeled Differently

The compasbin dataset has the lowest r-separateness. Its binary features are:

- sex:Female
- age:<21
- age:<23
- age:<26
- age:<46
- juvenile-felonies:=0
- juvenile-misdemeanors:=0
- juvenile-crimes:=0
- priors:=0
- priors:=1
- priors:2-3
- priors:>3

There are 852 people who are: male, age between 26 to 46, did not commit any juvenile felonies, misdemeanors, and crimes, and have more than 3 previous criminal conviction. These people will have the same feature vector while for their labels, 542 recidivate within two years while 310 people did not.

[6] `https://github.com/ustunb/risk-slim` and `https://github.com/chenhongge/RobustTrees`

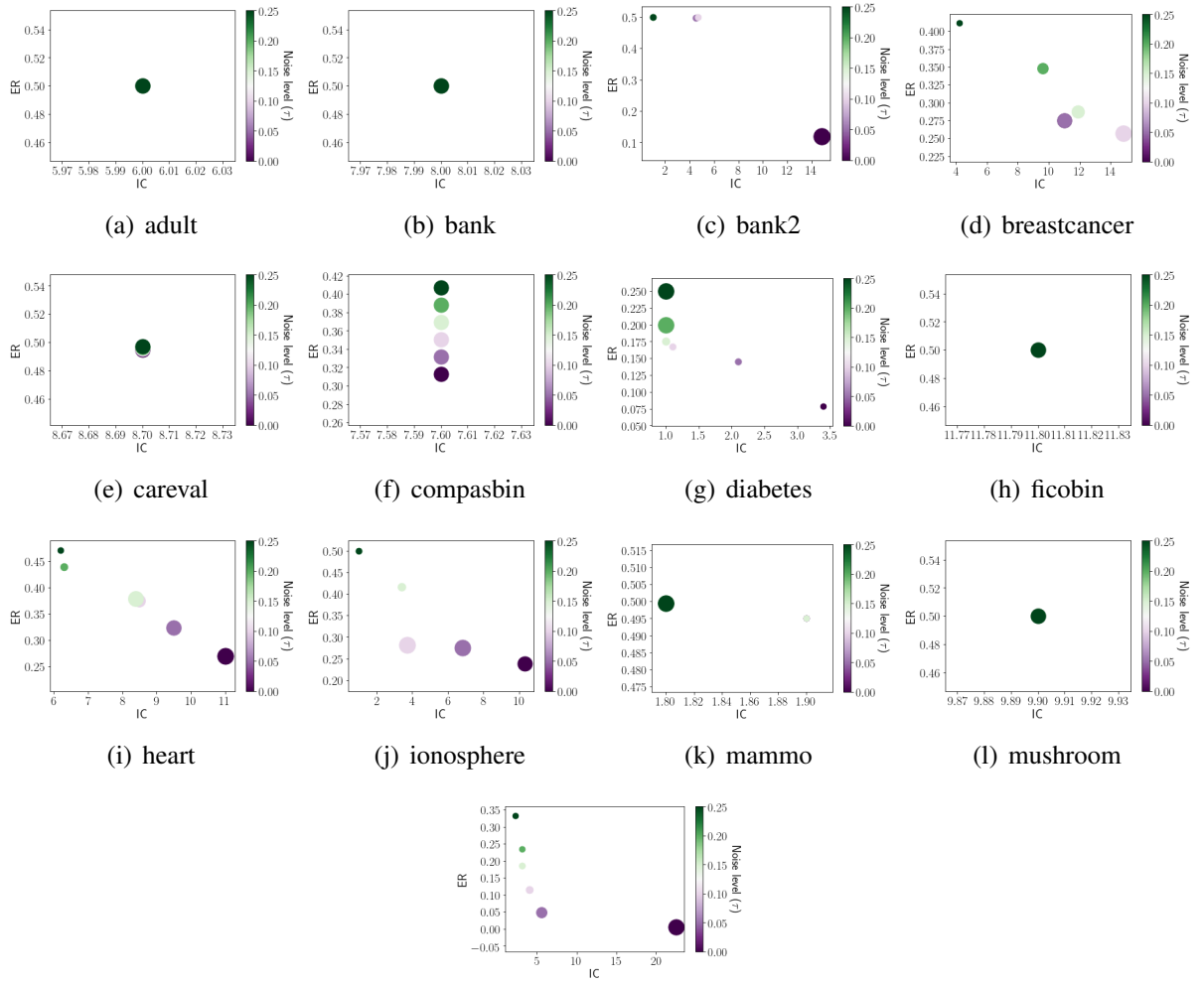
D.3.2 Relationship Between Explainability, Accuracy, and Robustness in BBM-RS

To understand the interaction between explainability, accuracy, and robustness, we measure these criteria of BBM-RS with different noise levels τ. The results are shown in Figure D.3. We observed that, by changing the noise level, that robustness and the explanation complexity go hand in hand. For higher noise levels, we have higher robustness and lower explanation complexity and accuracy. This shows that by making the model simpler, we can have better robustness and explainability while losing some accuracy.

D.3.3 Tradeoff Between Explanation Complexity and Accuracy for BBM-RS

To understand how explanation complexity affects accuracy, we first train a BBM-RS classifier. The learned BBM-RS classifier consists of T weak learners. We then measure the test accuracy of using only i weak learners for prediction, where $i = 1 \dots T$. Finally, we plot out the figure of accuracy versus explanation complexity (number of unique weak learners) in Figure D.4. Note that for the same explanation complexity, there may be more than one test accuracy. In this case, we show the highest test accuracy. In Figure D.4, we see that with the increase in explanation complexity, generally, the test accuracy also increases.

D.3.4 Local Interpretable Complexity



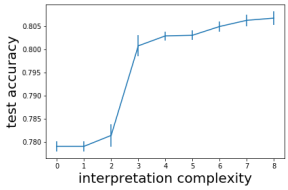
(a) adult (b) bank (c) bank2 (d) breastcancer

(e) careval (f) compasbin (g) diabetes (h) ficobin

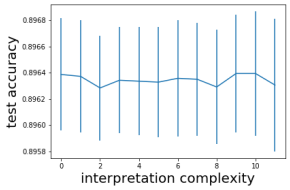
(i) heart (j) ionosphere (k) mammo (l) mushroom

(m) spambase

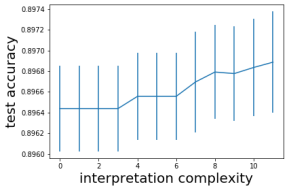
Figure D.3: The tradeoff between explainability and accuracy for BBM-RS. The size of the ball represents the accuracy.



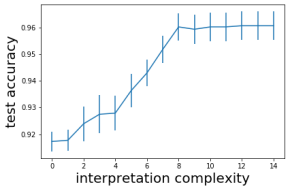
(a) adult



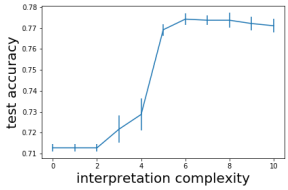
(b) bank



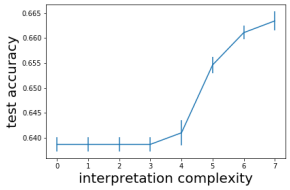
(c) bank2



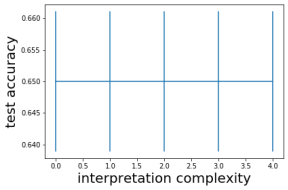
(d) breastcancer



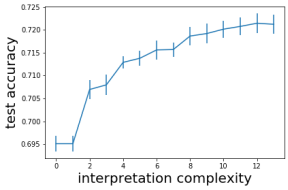
(e) careval



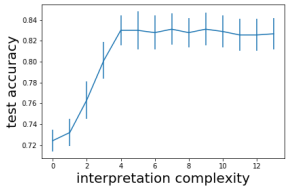
(f) compasbin



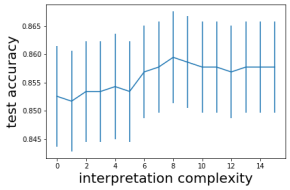
(g) diabetes



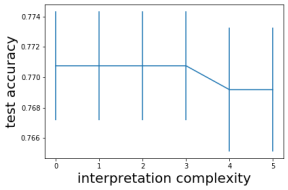
(h) ficobin



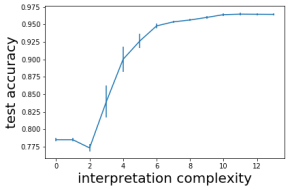
(i) heart



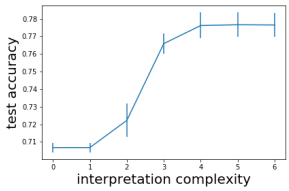
(j) ionosphere



(k) mammo



(l) mushroom



(m) spambase

Figure D.4: The tradeoff between explanation complexity and test accuracy for BBM-RS.

Table D.1: The comparison of BBM-RS with other interpretable models (with standard error).

	EC			
	DT	RobDT	Rudin's	BBM
adult	414.20 ± 5.66	287.90 ± 35.66	14.90 ± 1.46	**6.00 ± .60**
bank	30.70 ± .15	26.80 ± .20	8.90 ± .66	**8.00 ± 1.41**
bank2	30.00 ± .30	30.70 ± .15	13.80 ± 1.54	**4.50 ± 1.34**
breastcancer	15.20 ± 1.25	7.40 ± .60	**6.00 ± .00**	11.00 ± .89
careval	59.30 ± 2.22	28.20 ± .65	10.10 ± .97	**8.70 ± .47**
compasbin	67.80 ± 13.01	33.70 ± 3.05	**5.40 ± .22**	7.60 ± .16
diabetes	31.20 ± 6.96	27.90 ± 2.95	6.00 ± .00	**2.10 ± .53**
ficobin	30.60 ± .22	59.60 ± 29.82	**6.40 ± .16**	11.80 ± .65
heart	20.30 ± 1.60	13.60 ± .88	11.90 ± 1.46	**9.50 ± .82**
ionosphere	11.30 ± .98	8.60 ± .76	17.90 ± 3.14	**6.80 ± 1.96**
mammo	27.40 ± 5.09	12.40 ± .65	7.20 ± .65	**1.90 ± .60**
mushroom	10.80 ± .25	**9.10 ± .10**	23.80 ± 1.50	9.90 ± .89
spambase	153.90 ± 8.51	72.30 ± 2.89	29.50 ± .76	**5.60 ± .48**
	test accuracy			
adult	**0.83 ± .00**	**0.83 ± .00**	0.82 ± .00	0.81 ± .00
bank	**0.90 ± .00**	**0.90 ± .00**	**0.90 ± .00**	**0.90 ± .00**
bank2	**0.91 ± .00**	0.90 ± .00	0.90 ± .00	0.90 ± .00
breastcancer	0.94 ± .00	0.94 ± .01	**0.96 ± .00**	**0.96 ± .01**
careval	**0.97 ± .00**	0.96 ± .00	0.91 ± .01	0.77 ± .00
compasbin	**0.67 ± .00**	**0.67 ± .00**	0.65 ± .00	0.66 ± .00
diabetes	0.74 ± .01	0.73 ± .01	**0.76 ± .01**	0.65 ± .01
ficobin	0.71 ± .00	0.71 ± .00	0.71 ± .00	**0.72 ± .00**
heart	0.76 ± .01	0.79 ± .01	**0.82 ± .01**	**0.82 ± .01**
ionosphere	0.89 ± .01	**0.92 ± .01**	0.88 ± .01	0.86 ± .01
mammo	**0.79 ± .00**	**0.79 ± .00**	**0.79 ± .00**	0.77 ± .00
mushroom	**1.00 ± .00**	**1.00 ± .00**	**1.00 ± .00**	0.97 ± .00
spambase	**0.92 ± .00**	0.87 ± .00	0.88 ± .00	0.79 ± .01
	ER			
adult	**0.50 ± .00**	**0.50 ± .00**	0.12 ± .02	**0.50 ± .00**
bank	**0.50 ± .00**	**0.50 ± .00**	0.20 ± .03	**0.50 ± .00**
bank2	0.12 ± .01	0.18 ± .02	0.10 ± .01	**0.50 ± .00**
breastcancer	0.23 ± .01	**0.29 ± .01**	0.28 ± .00	0.27 ± .01
careval	**0.50 ± .00**	**0.50 ± .00**	0.19 ± .02	**0.50 ± .00**
compasbin	**0.50 ± .00**	**0.50 ± .00**	0.15 ± .01	0.33 ± .01
diabetes	0.08 ± .01	0.08 ± .00	0.09 ± .00	**0.15 ± .05**
ficobin	**0.50 ± .00**	**0.50 ± .00**	0.22 ± .01	**0.50 ± .00**
heart	0.23 ± .02	0.31 ± .02	0.14 ± .01	**0.32 ± .02**
ionosphere	0.15 ± .01	0.25 ± .01	0.07 ± .01	**0.28 ± .01**
mammo	0.47 ± .01	**0.50 ± .00**	0.21 ± .02	**0.50 ± .00**
mushroom	**0.50 ± .00**	**0.50 ± .00**	0.10 ± .01	**0.50 ± .00**
spambase	0.00 ± .00	0.04 ± .00	0.02 ± .00	**0.05 ± .00**

Table D.2: Interpretable complexity. DT measured by depth

	DT	RobDT	Rudin's	BBM
adult	$10.00 \pm .00$	12.50 ± 1.07	14.90 ± 1.46	$\mathbf{6.00 \pm .60}$
bank	$\mathbf{5.00 \pm .00}$	$6.00 \pm .00$	$8.90 \pm .66$	8.00 ± 1.41
bank2	$5.00 \pm .00$	$6.00 \pm .00$	13.80 ± 1.54	$\mathbf{4.50 \pm 1.34}$
breastcancer	$6.00 \pm .54$	$\mathbf{5.20 \pm .20}$	$6.00 \pm .00$	$11.00 \pm .89$
careval	$12.30 \pm .54$	$11.40 \pm .16$	$10.10 \pm .97$	$\mathbf{8.70 \pm .47}$
compasbin	$7.40 \pm .81$	$7.90 \pm .53$	$\mathbf{5.40 \pm .22}$	$7.60 \pm .16$
diabetes	$6.00 \pm .67$	$7.50 \pm .76$	$6.00 \pm .00$	$\mathbf{2.10 \pm .53}$
ficobin	$\mathbf{5.00 \pm .00}$	7.00 ± 1.00	$6.40 \pm .16$	$11.80 \pm .65$
heart	$\mathbf{6.00 \pm .52}$	$6.10 \pm .10$	11.90 ± 1.46	$9.50 \pm .82$
ionosphere	$\mathbf{6.00 \pm .42}$	$7.90 \pm .59$	17.90 ± 3.14	6.80 ± 1.96
mammo	$5.60 \pm .60$	$6.20 \pm .13$	$7.20 \pm .65$	$\mathbf{1.90 \pm .60}$
mushroom	$\mathbf{5.80 \pm .25}$	$6.00 \pm .00$	23.80 ± 1.50	$9.90 \pm .89$
spambase	17.40 ± 1.36	$17.60 \pm .67$	$29.50 \pm .76$	$\mathbf{5.60 \pm .48}$

Appendix E

Probing Predictions on OOD Images via Nearest Categories

E.1 Detailed Experiment Setups

The experiments are performed on 6 NVIDIA GeForce RTX 2080 Ti and 2 RTX 3080 GPUs located on three servers. Two of the servers have Intel Core i9 9940X and 128GB of RAM, and the other one has AMD Threadripper 3960X and 256GB of RAM. We compute nearest neighbors using FAISS[1] [104], and all neural networks are implemented under the PyTorch framework[2] [168]

Algorithm implementations. For C&W algorithm [39], we use the implementation by For TRADES [244], we also use the implementation From the original author[3].

Datasets. All datasets used in our paper can be found in publicly available urls. MNIST can be found in this url[4], CIFAR10 and CIFAR100 can be found in this url[5], ImgNet can be found

[1] code and license can be found in `https://github.com/facebookresearch/faiss`
[2] code and license can be found in `https://github.com/pytorch/pytorch`
[3] code and license can be found in `https://github.com/yaodongyu/TRADES`
[4] `http://yann.lecun.com/exdb/mnist/`
[5] `https://www.cs.toronto.edu/~kriz/cifar.html`

in this url[6].

Architechtures. We consider the convolutional neural network (CNN)[7], wider residual network (WRN-40-10) [242], ResNet50 [92] for our experiments in the pixel space.

Optimizers. We consider stochastic gradient descent (SGD) and Adam [115] as the optimizers.

MNIST setup. We use the CNN used by Zhang et al. [244] for training neural networks in the pixel space. The learning rate is decreased by a factor of 0.1 on the 40-th, 50-th, and 60-th epoch. We use the output of the last convolutional CNN output as the extracted feature.

CIFAR10, CIFAR100, ImgNet100 setup. For CIFAR10 and CIFAR100, we use Wider ResNet (WRN-40-10) [242] for training neural networks in the pixel space. For ImgNet100, we use ResNet50 [92] for training neural networks in the pixel space. The learning rate is decreased by a factor of 0.1 on the 40-th, 50-th, and 60-th epoch. For ImgNet100, we normalize the data by subtracting the mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225).

Table E.1: Experimental setup for training in the pixel space. No weight decay is applied.

dataset	MNIST	CIFAR10	CIFAR100	ImgNet100
network structure	CNN	WRN-40-10	WRN-40-10	ResNet50
optimizer	SGD	Adam	Adam	Adam
batch size	128	64	64	128
momentum	0.9	-	-	-
epochs	70	70	70	70
initial learning rate	0.01	0.01	0.01	0.01
# train examples	60000	50000	50000	126689
# test examples	10000	10000	10000	5000
# classes	10	10	20	100

Adversarial attack algorithms. For the adversarial attack algorithms used to find the closest adversarial examples, we use a mixture of projected gradient descent (PGD) [144], Brendel

[6] https://www.image-net.org/

[7] CNN is retrieved from the public repository of TRADES [244] https://github.com/yaodongyu/TRADES/blob/master/models/small_cnn.py

Bethge attack [33], boundary attack [32], multi-targeted attack [125], Sign-Opt [51] and C&W algorithm [39].

Setups for experiments in the feature space

Architechtures. For MNIST, CIFAR10, and CIFAR100, we train a multi-layer-perceptron (MLP) with two hidden layers each with 256 neurons and ReLU as the activation function in the feature space. For ImgNet100, we train an MLP with two hidden layers each with 1024 neurons and ReLU as the activation function in the feature space. For all four datasets, we use SGD as the optimizer with an initial learning rate of 0.01 and a momentum of 0.9.

E.2 Additional Experiment Results

E.2.1 NCG Accuracies

Table E.2 shows the test accuracies of the 1-NN classifiers in the feature space of 12 different datasets. Tables E.3 to E.6 extends Table 6.2 with all datasets. We see that the 1-NN classifiers are actually performing very well (close) in the feature space.

Table E.2: The test accuracy of a 1-nearest neighbor classifier in the feature space 12 different datasets.

M-0	M-4	M-9	C10-0	C10-4	C10-9	C100-0	C100-4	C100-9	I-0	I-1	I-2
0.99	0.99	0.99	0.89	0.88	0.88	0.73	0.73	0.71	0.14	0.14	0.14

E.2.2 Ablation Study

Different architectures

We repeat the experiment with a different network architecture – DenseNet161 [98]. Their training, testing, and NCG accuracies are shown in Table E.7.

Table E.3: The train, test, and NCG accuracies of 10 MNIST datasets and 5 training methods in the pixel space.

		natural	AT(2)	TRADES(2)	TRADES(4)	TRADES(8)
MNIST-wo0	train acc.	1.000	0.993	0.987	0.954	0.997
	test acc.	0.995	0.990	0.985	0.956	0.995
	NCG acc.	0.390	0.457	0.457	0.485	0.402
MNIST-wo1	train acc.	1.000	0.994	0.987	0.975	0.997
	test acc.	0.995	0.991	0.987	0.974	0.994
	NCG acc.	0.273	0.451	0.355	0.528	0.259
MNIST-wo2	train acc.	1.000	0.993	0.988	0.958	0.997
	test acc.	0.994	0.990	0.987	0.962	0.994
	NCG acc.	0.402	0.532	0.529	0.520	0.452
MNIST-wo3	train acc.	1.000	0.994	0.989	0.962	0.997
	test acc.	0.995	0.992	0.988	0.964	0.994
	NCG acc.	0.564	0.659	0.667	0.592	0.538
MNIST-wo4	train acc.	1.000	0.994	0.988	0.963	0.997
	test acc.	0.995	0.991	0.987	0.966	0.995
	NCG acc.	0.760	0.766	0.810	0.758	0.749
MNIST-wo5	train acc.	1.000	0.993	0.988	0.965	0.997
	test acc.	0.995	0.990	0.987	0.965	0.995
	NCG acc.	0.505	0.611	0.618	0.616	0.537
MNIST-wo6	train acc.	1.000	0.993	0.987	0.959	0.997
	test acc.	0.995	0.991	0.987	0.962	0.995
	NCG acc.	0.515	0.551	0.556	0.505	0.538
MNIST-wo7	train acc.	1.000	0.994	0.989	0.962	0.997
	test acc.	0.995	0.992	0.990	0.967	0.994
	NCG acc.	0.507	0.672	0.703	0.713	0.594
MNIST-wo8	train acc.	1.000	0.993	0.987	0.966	0.997
	test acc.	0.994	0.990	0.987	0.966	0.995
	NCG acc.	0.416	0.493	0.497	0.491	0.446
MNIST-wo9	train acc.	1.000	0.996	0.992	0.962	0.997
	test acc.	0.996	0.994	0.992	0.964	0.995
	NCG acc.	0.577	0.714	0.691	0.703	0.660

Pretrainde Models

To verify that our observations can also be observed by models trained by others, we downloaded pretrained models from `https://github.com/MadryLab/robustness/tree/master/robustness` by Engstrom et al. [65]. Table E.8 shows the training and testing accuracies of their

Table E.4: The train, test, and NCG accuracies of 9 different variations of CIFAR10, CIFAR100, and ImgNet100 datasets and 5 training methods in the pixel space.

		natural	AT(2)	TRADES(2)	TRADES(4)	TRADES(8)
CIFAR10-wo0	train acc.	1.000	0.999	0.992	0.870	0.878
	test acc.	0.898	0.729	0.716	0.660	0.761
	NCG acc.	0.355	0.494	0.492	0.520	0.483
CIFAR10-wo4	train acc.	1.000	1.000	0.990	0.874	0.508
	test acc.	0.886	0.754	0.742	0.700	0.485
	NCG acc.	0.222	0.361	0.333	0.331	0.289
CIFAR10-wo9	train acc.	1.000	1.000	0.992	0.948	0.778
	test acc.	0.885	0.725	0.712	0.732	0.641
	NCG acc.	0.145	0.212	0.192	0.247	0.245
CIFAR100-wo0	train acc.	1.000	0.998	0.995	0.943	0.902
	test acc.	0.741	0.554	0.547	0.576	0.607
	NCG acc.	0.175	0.240	0.252	0.252	0.206
CIFAR100-wo4	train acc.	1.000	0.998	0.995	0.857	0.859
	test acc.	0.743	0.544	0.543	0.492	0.553
	NCG acc.	0.137	0.192	0.191	0.187	0.185
CIFAR100-wo9	train acc.	1.000	0.996	0.995	0.950	0.527
	test acc.	0.727	0.547	0.537	0.585	0.431
	NCG acc.	0.222	0.353	0.412	0.427	0.465
ImgNet100-wo0	train acc.	1.000	0.999	0.994	0.983	0.704
	test acc.	0.529	0.417	0.393	0.354	0.320
	NCG acc.	0.033	0.044	0.041	0.054	0.067
ImgNet100-wo1	train acc.	1.000	0.999	0.995	0.972	0.783
	test acc.	0.534	0.414	0.385	0.356	0.316
	NCG acc.	0.047	0.049	0.051	0.061	0.072
ImgNet100-wo2	train acc.	1.000	0.999	0.994	0.971	0.695
	test acc.	0.537	0.394	0.388	0.353	0.320
	NCG acc.	0.027	0.028	0.033	0.044	0.049

models.

Corrupted data. For models in the features space, we follow the same setup as in the feature space of CIFAR10, which still trains a multi-layer perceptron on the CNN feature space, but in the feature space of the pretrained model. Table E.9 shows the comparison of robust and naturally trained models. From the table, we can see that the robust models, in general, have higher NCG accuracy than the naturally trained models when the robust radius r is larger than

Table E.5: The train, test, and NCG accuracies of 10 MNIST datasets and 5 training methods in the feature space.

		natural	AT(2)	TRADES(2)	TRADES(4)	TRADES(8)
MNIST-wo0	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	0.99	0.99
	NCG acc.	0.28	0.32	0.39	0.49	0.55
MNIST-wo1	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	0.99	0.99
	NCG acc.	0.14	0.21	0.27	0.50	0.51
MNIST-wo2	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	0.99	1.00
	NCG acc.	0.41	0.46	0.53	0.59	0.62
MNIST-wo3	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	1.00	0.99
	NCG acc.	0.68	0.71	0.73	0.73	0.74
MNIST-wo4	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	1.00	1.00
	NCG acc.	0.78	0.73	0.77	0.81	0.86
MNIST-wo5	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	1.00	1.00	0.99
	NCG acc.	0.61	0.63	0.65	0.68	0.69
MNIST-wo6	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	1.00	1.00	1.00	1.00
	NCG acc.	0.54	0.58	0.60	0.65	0.66
MNIST-wo7	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	1.00	1.00	1.00
	NCG acc.	0.53	0.54	0.61	0.68	0.67
MNIST-wo8	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	0.99	0.99	1.00	0.99
	NCG acc.	0.46	0.47	0.51	0.56	0.59
MNIST-wo9	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.99	1.00	1.00	1.00	1.00
	NCG acc.	0.61	0.71	0.71	0.73	0.79

0.25. Table E.10 shows the test accuracy, NCG accuracy, and the test accuracy conditioned on whether the example is considered correct under NCG accuracy (NCG correct or not).

Table E.6: The train, test, and NCG accuracies of nine different variations of CIFAR10, CIFAR100, and ImgNet100 datasets and five training methods in the feature space. We use different radius for AT since not all converge well when the radius is large ($r = 2$) For CIFAR10 and CIFAR100, we use AT(1); for ImgNet100, we use AT(.5).

		natural	AT(.5)/(1)	TRADES(2)	TRADES(4)	TRADES(8)
CIFAR10-wo0	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.89	0.89	0.89	0.90	0.90
	NCG acc.	0.80	0.83	0.81	0.83	0.83
CIFAR10-wo4	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.88	0.88	0.88	0.89	0.88
	NCG acc.	0.82	0.84	0.82	0.85	0.85
CIFAR10-wo9	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.88	0.88	0.88	0.89	0.89
	NCG acc.	0.84	0.89	0.83	0.88	0.87
CIFAR100-wo0	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.72	0.73	0.73	0.74	0.74
	NCG acc.	0.63	0.70	0.69	0.68	0.68
CIFAR100-wo4	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.72	0.73	0.73	0.74	0.74
	NCG acc.	0.69	0.74	0.75	0.73	0.74
CIFAR100-wo9	train acc.	1.00	1.00	1.00	1.00	1.00
	test acc.	0.70	0.72	0.72	0.73	0.73
	NCG acc.	0.66	0.74	0.72	0.71	0.71
ImgNet100-wo0	train acc.	0.99	0.57	0.33	0.98	0.98
	test acc.	0.22	0.25	0.26	0.26	0.26
	NCG acc.	0.11	0.16	0.15	0.12	0.13
ImgNet100-wo1	train acc.	1.00	0.56	0.32	0.98	0.98
	test acc.	0.22	0.24	0.27	0.26	0.25
	NCG acc.	0.13	0.15	0.18	0.14	0.15
ImgNet100-wo2	train acc.	1.00	0.60	0.33	0.98	0.98
	test acc.	0.22	0.25	0.26	0.26	0.26
	NCG acc.	0.11	0.15	0.15	0.14	0.14

E.2.3 Histograms of the Empirical Robust Radius and OOD Distance

Here we present the histogram of the empirical robust radius and OOD distance for other algorithms and datasets. Figures E.1 to E.3 show the results for MNIST, CIFAR10, CIFAR100, and ImgNet100 in the pixel space. Figures E.4 to E.6 show the results for MNIST, CIFAR10, CIFAR100, and ImgNet100 in the feature space.

For the MNIST histograms in the feature space, we see that the empirical robust radius

Table E.7: Results with DenseNet161 on CIFAR10 and CIFAR100.

		natural	AT(2)	TRADES(2)
CIFAR10-wo0	train acc.	1.000	0.781	0.876
	test acc.	0.839	0.637	0.640
	NCG acc.	0.342	0.487	0.521
CIFAR100-wo0	train acc.	1.000	0.886	0.557
	test acc.	0.608	0.500	0.441
	NCG acc.	0.173	0.225	0.271

Table E.8: The training and testing accuracies of the Engstrom et al. [65]'s pretrained models on CIFAR10.

	natural	AT(.25)	AT(.5)	AT(1.0)
trn acc.	1.00	0.97	0.98	0.86
tst acc.	0.95	0.93	0.91	0.82

have smaller number compared with the counts for OOD distance. That is because there are many OOD examples that have these few training examples as the closest training example.

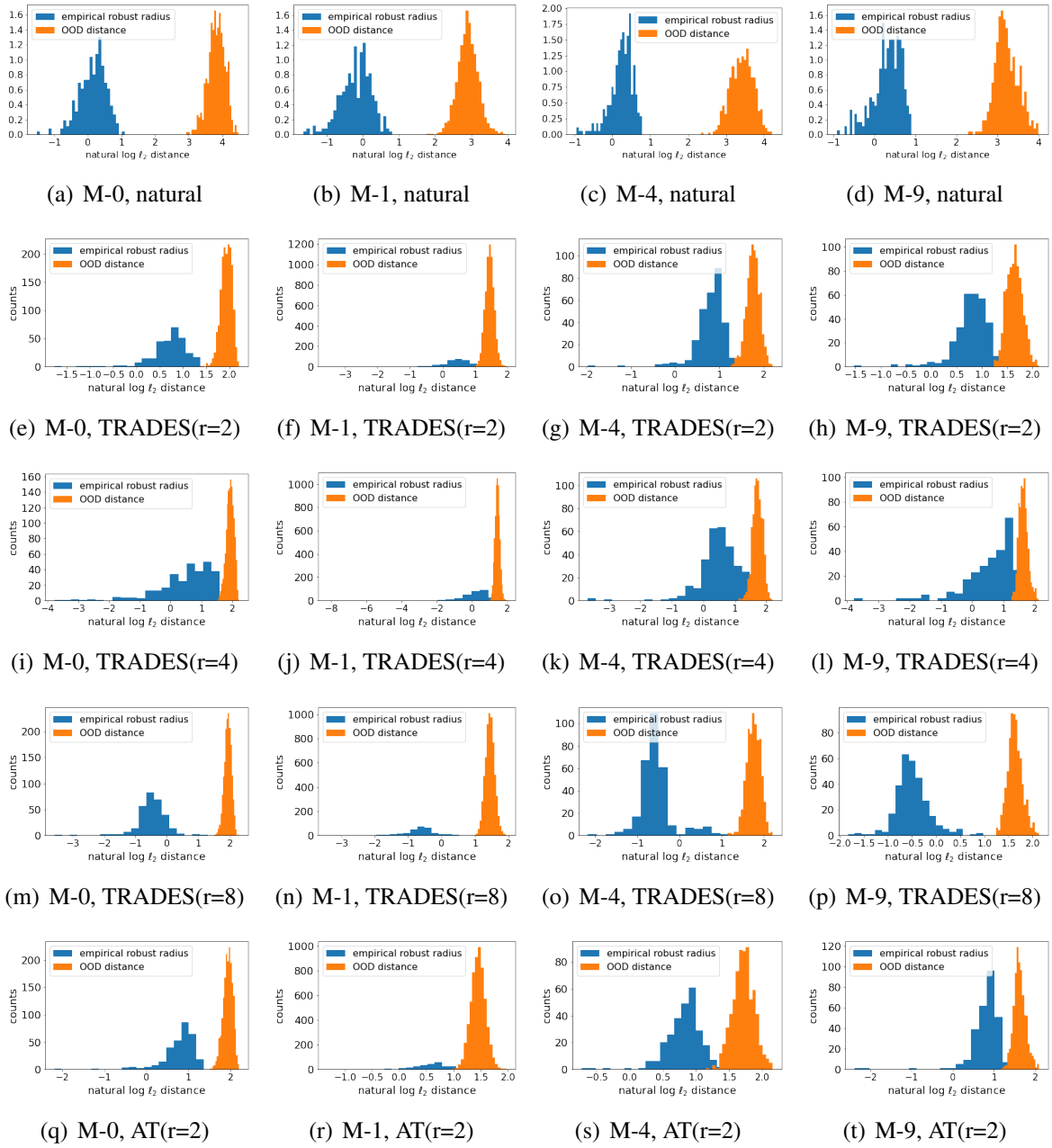
Tables E.11 to E.14 show the average empirical robust radius, average OOD distance, portion of OOD examples covered by the robust norm ball of its closest training example and the NCG accuracy (in the pixel and feature space of M, C10, C100, and I). From the table, we can see that the portion of OOD examples covered by the robust norm ball of its closest training example are very low in general, regardless of the NCG accuracy. This rejects the hypothesis that the robust methods enforce the neural network to be locally smooth in a ball of radius r; if the OOD inputs are closer than r from their nearest training example, then they would get classified accordingly. Next, we test if this is the case by measuring the distances between the OOD inputs and their closest training examples.

Table E.9: In both pixel and feature space, among the 90 corrupted sets for CIFAR10, the first columns shows the number of robust models that have an NCG accuracy higher than naturally trained network. The second and third column shows the average difference and ratio of the NCG accuracy of the robust models and naturally trained networks (average over the NCG accuracies on the 90 corrupted sets).

		robust $>$ natural counts	difference	ratio
			pixel	
CIFAR10	AT(0.25)	51/90	0.00 ± 0.05	1.14 ± 0.04
	AT(0.5)	86/90	0.14 ± 0.10	3.27 ± 0.55
	AT(1.0)	88/90	0.18 ± 0.06	3.09 ± 0.22
			feature	
CIFAR10	AT(1.0)	70/90	0.00 ± 0.00	1.01 ± 0.00
	TRADES(2)	55/90	0.00 ± 0.00	1.00 ± 0.00
	TRADES(4)	52/90	0.00 ± 0.00	1.00 ± 0.00
	TRADES(8)	55/90	0.00 ± 0.00	1.00 ± 0.00

Table E.10: The test accuracy, NCG accuracy, and the test accuracy conditioned on the NCG correctness of Engstrom et al. [65]'s pretrained model on the Gaussian noise corrupted data.

	model	natural				AT(1)			
dataset	level	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.
					pixel				
C10	1	0.52	0.50	0.68	0.13	0.21	0.17	0.31	0.30
	2	0.37	0.35	0.49	0.12	0.20	0.16	0.31	0.29
	3	0.28	0.26	0.38	0.13	0.20	0.16	0.30	0.29
	4	0.25	0.24	0.33	0.14	0.20	0.16	0.30	0.28
	5	0.23	0.21	0.32	0.14	0.20	0.16	0.30	0.28
					feature				
C10	1	0.82	0.44	0.85	0.95	0.82	0.42	0.83	0.97
	2	0.67	0.38	0.70	0.91	0.66	0.41	0.68	0.95
	3	0.49	0.31	0.52	0.86	0.48	0.30	0.49	0.93
	4	0.42	0.29	0.44	0.85	0.41	0.31	0.42	0.92
	5	0.36	0.27	0.37	0.84	0.35	0.27	0.35	0.92



(a) M-0, natural (b) M-1, natural (c) M-4, natural (d) M-9, natural

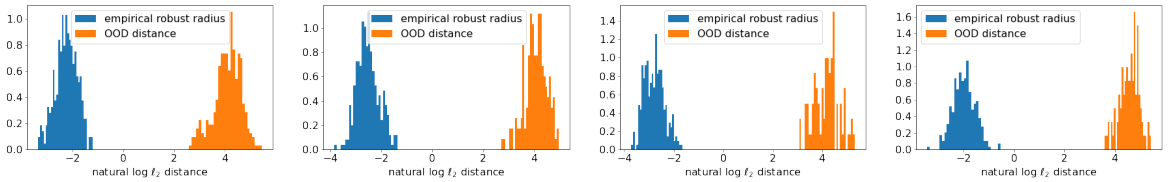
(e) M-0, TRADES(r=2) (f) M-1, TRADES(r=2) (g) M-4, TRADES(r=2) (h) M-9, TRADES(r=2)

(i) M-0, TRADES(r=4) (j) M-1, TRADES(r=4) (k) M-4, TRADES(r=4) (l) M-9, TRADES(r=4)

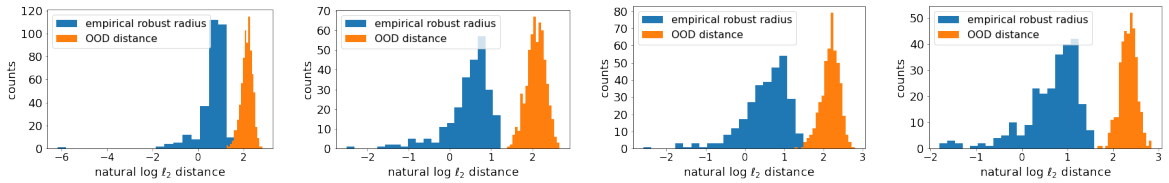
(m) M-0, TRADES(r=8) (n) M-1, TRADES(r=8) (o) M-4, TRADES(r=8) (p) M-9, TRADES(r=8)

(q) M-0, AT(r=2) (r) M-1, AT(r=2) (s) M-4, AT(r=2) (t) M-9, AT(r=2)

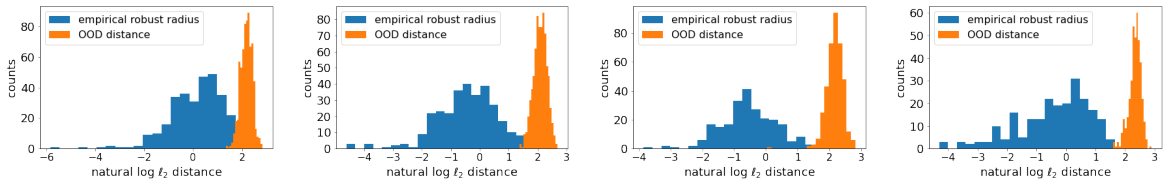
Figure E.1: The histograms of the empirical robust radius and OOD distance for networks trained on MNIST-wo0 (M-0), MNIST-wo1 (M-1), MNIST-wo4 (M-4), and MNIST-wo9 (M-9) in the pixel space.



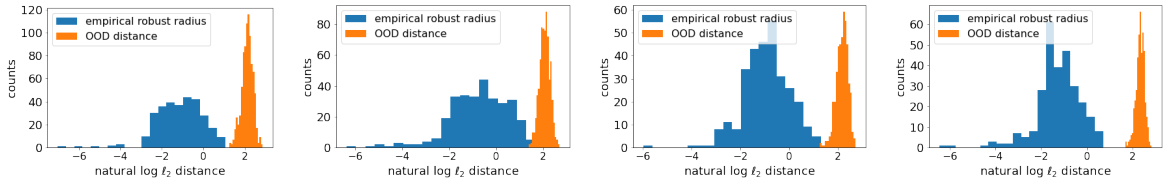
(a) C10-0, natural (b) C10-4, natural (c) C100-0, natural (d) C100-4, natural



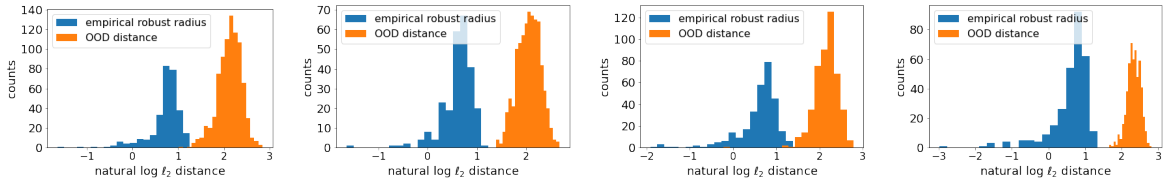
(e) C10-0, TRADES(r=2) (f) C10-4, TRADES(r=2) (g) C100-0, TRADES(r=2) (h) C100-4, TRADES(r=2)



(i) C10-0, TRADES(r=4) (j) C10-4, TRADES(r=4) (k) C100-0, TRADES(r=4) (l) C100-4, TRADES(r=4)



(m) C10-0, TRADES(r=8) (n) C10-4, TRADES(r=8) (o) C100-0, TRADES(r=8) (p) C100-4, TRADES(r=8)

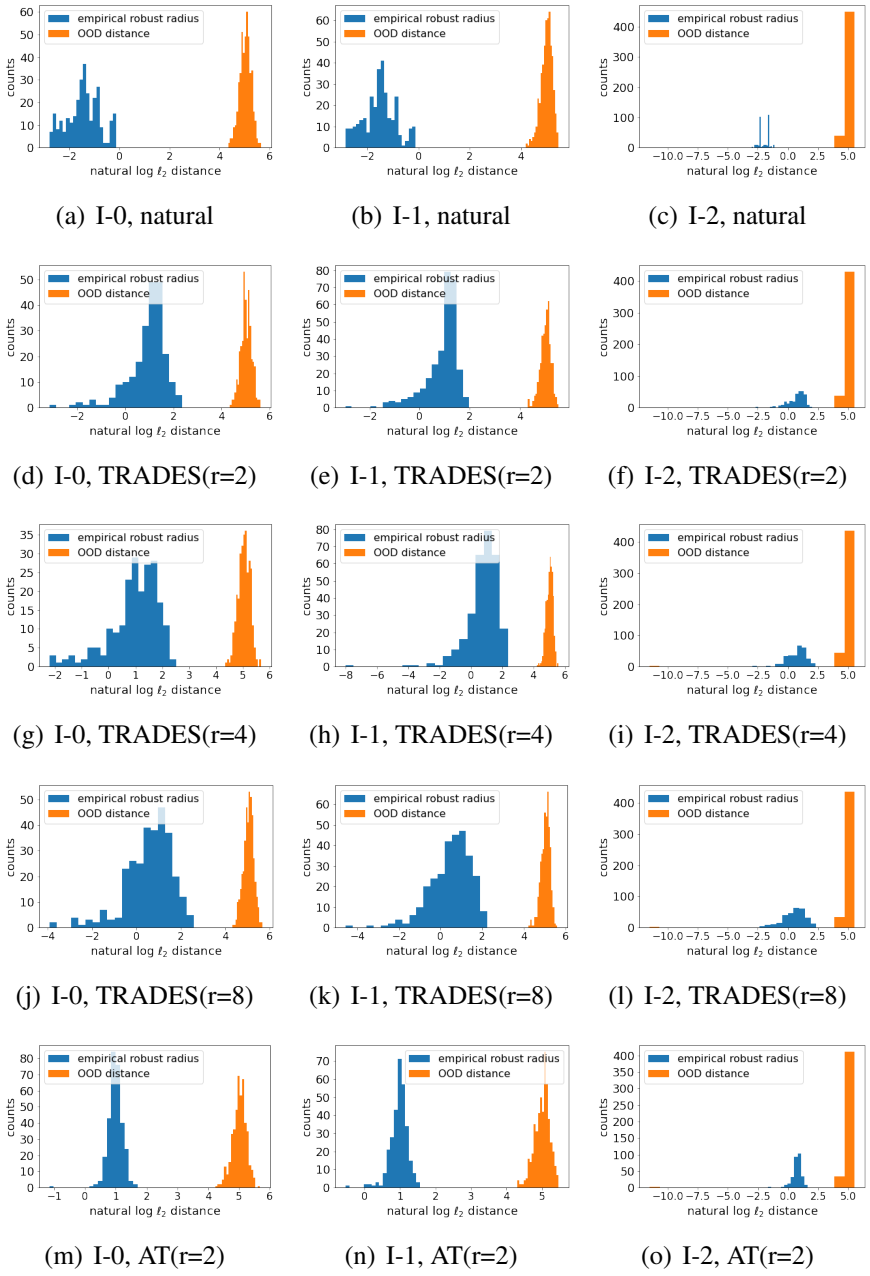


(q) C10-0, AT(r=2) (r) C10-4, AT(r=2) (s) C100-0, AT(r=2) (t) C100-4, AT(r=2)

Figure E.2: The histograms of the empirical robust radius and OOD distance for networks trained on CIFAR10-wo0 (C10-0), CIFAR10-wo4 (C10-4), CIFAR100-wo0 (C100-4), and CIFAR100-wo4 (C100-4) in the pixel space.

E.2.4 Additional Figures on NCG Accuracy and the Distance to the Closest Training Example

Figures E.7 and E.8 shows the NCG accuracy and the distance to the closest training example for MNIST, CIFAR10, and CIFAR100 in both pixel and feature space. We can see that,



(a) I-0, natural (b) I-1, natural (c) I-2, natural

(d) I-0, TRADES(r=2) (e) I-1, TRADES(r=2) (f) I-2, TRADES(r=2)

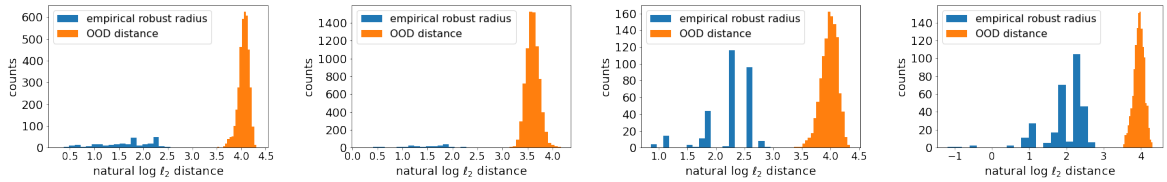
(g) I-0, TRADES(r=4) (h) I-1, TRADES(r=4) (i) I-2, TRADES(r=4)

(j) I-0, TRADES(r=8) (k) I-1, TRADES(r=8) (l) I-2, TRADES(r=8)

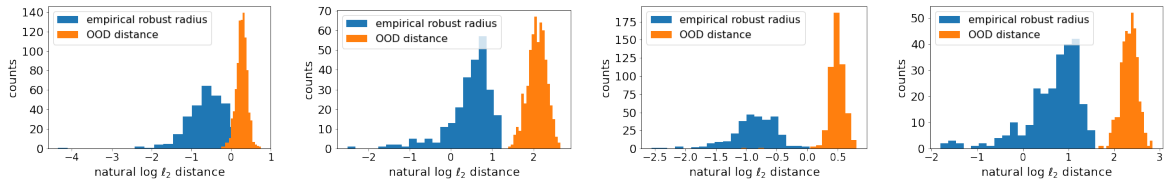
(m) I-0, AT(r=2) (n) I-1, AT(r=2) (o) I-2, AT(r=2)

Figure E.3: The histograms of the empirical robust radius and OOD distance for networks trained on ImgNet100-wo0(I-0), ImgNet100-wo0(I-1), and ImgNet100-wo0(I-2) in the pixel space.

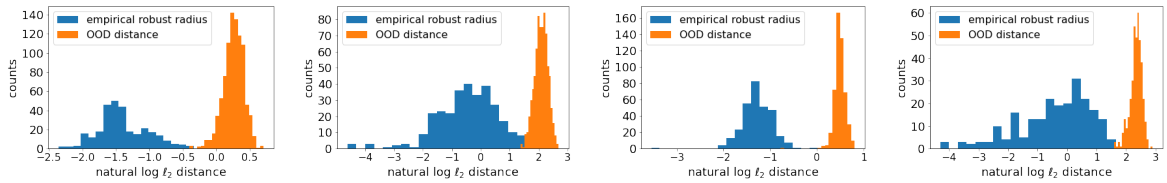
in general, the NCG accuracy is higher when in- and out-of-distribution examples are closer to each other.



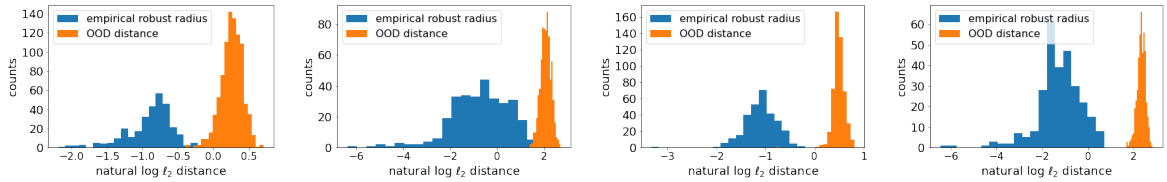
(a) M-0, natural (b) M-1, natural (c) M-4, natural (d) M-9, natural



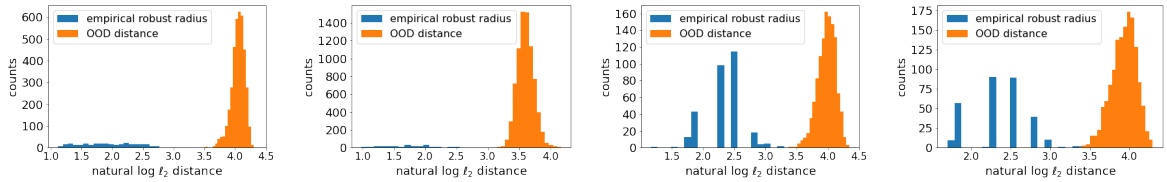
(e) M-0, TRADES(r=2) (f) M-1, TRADES(r=2) (g) M-4, TRADES(r=2) (h) M-9, TRADES(r=2)



(i) M-0, TRADES(r=4) (j) M-1, TRADES(r=4) (k) M-4, TRADES(r=4) (l) M-9, TRADES(r=4)

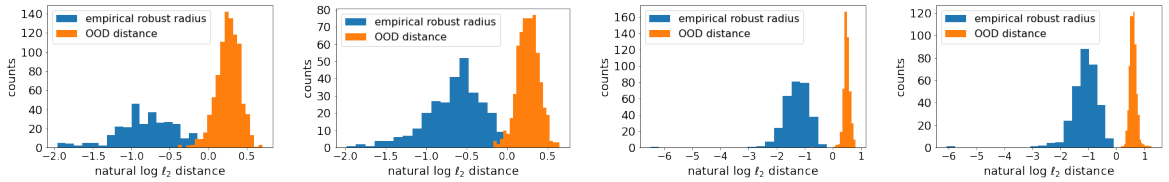


(m) M-0, TRADES(r=8) (n) M-1, TRADES(r=8) (o) M-4, TRADES(r=8) (p) M-9, TRADES(r=8)

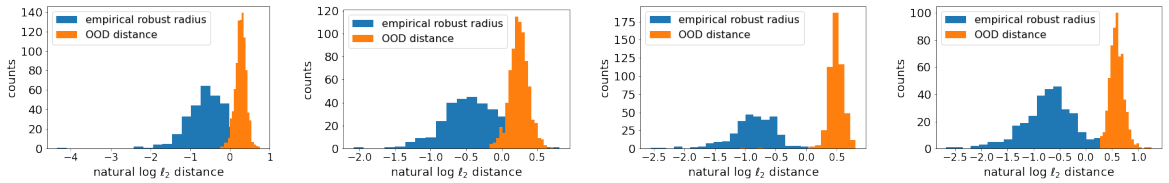


(q) M-0, AT(r=2) (r) M-1, AT(r=2) (s) M-4, AT(r=2) (t) M-9, AT(r=2)

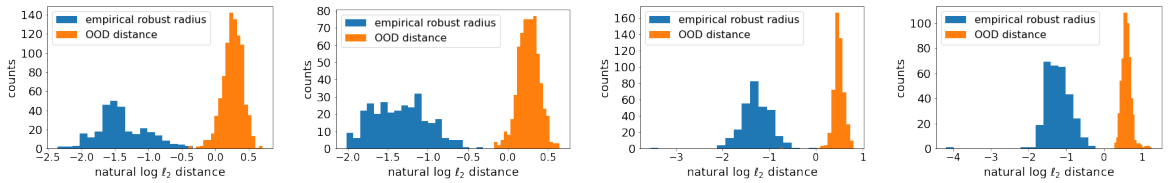
Figure E.4: The histograms of the empirical robust radius and OOD distance for networks trained on MNIST-wo0 (M-0), MNIST-wo1 (M-1), MNIST-wo4 (C100-4), and MNIST-wo9 (C100-9) in the feature space.



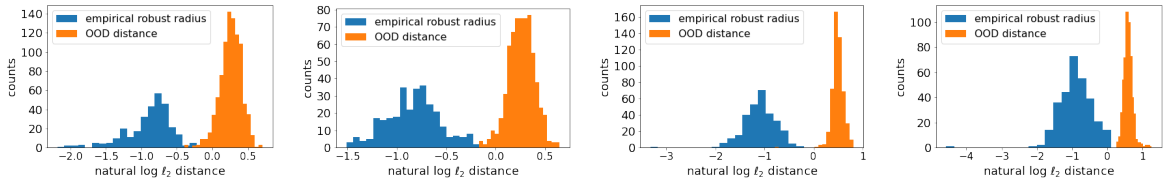
(a) C10-0, natural (b) C10-4, natural (c) C100-0, natural (d) C100-4, natural



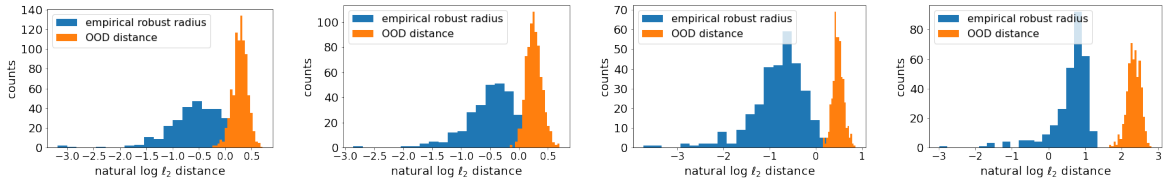
(e) C10-0, TRADES(r=2) (f) C10-4, TRADES(r=2) (g) C100-0, TRADES(r=2) (h) C100-4, TRADES(r=2)



(i) C10-0, TRADES(r=4) (j) C10-4, TRADES(r=4) (k) C100-0, TRADES(r=4) (l) C100-4, TRADES(r=4)



(m) C10-0, TRADES(r=8) (n) C10-4, TRADES(r=8) (o) C100-0, TRADES(r=8) (p) C100-4, TRADES(r=8)

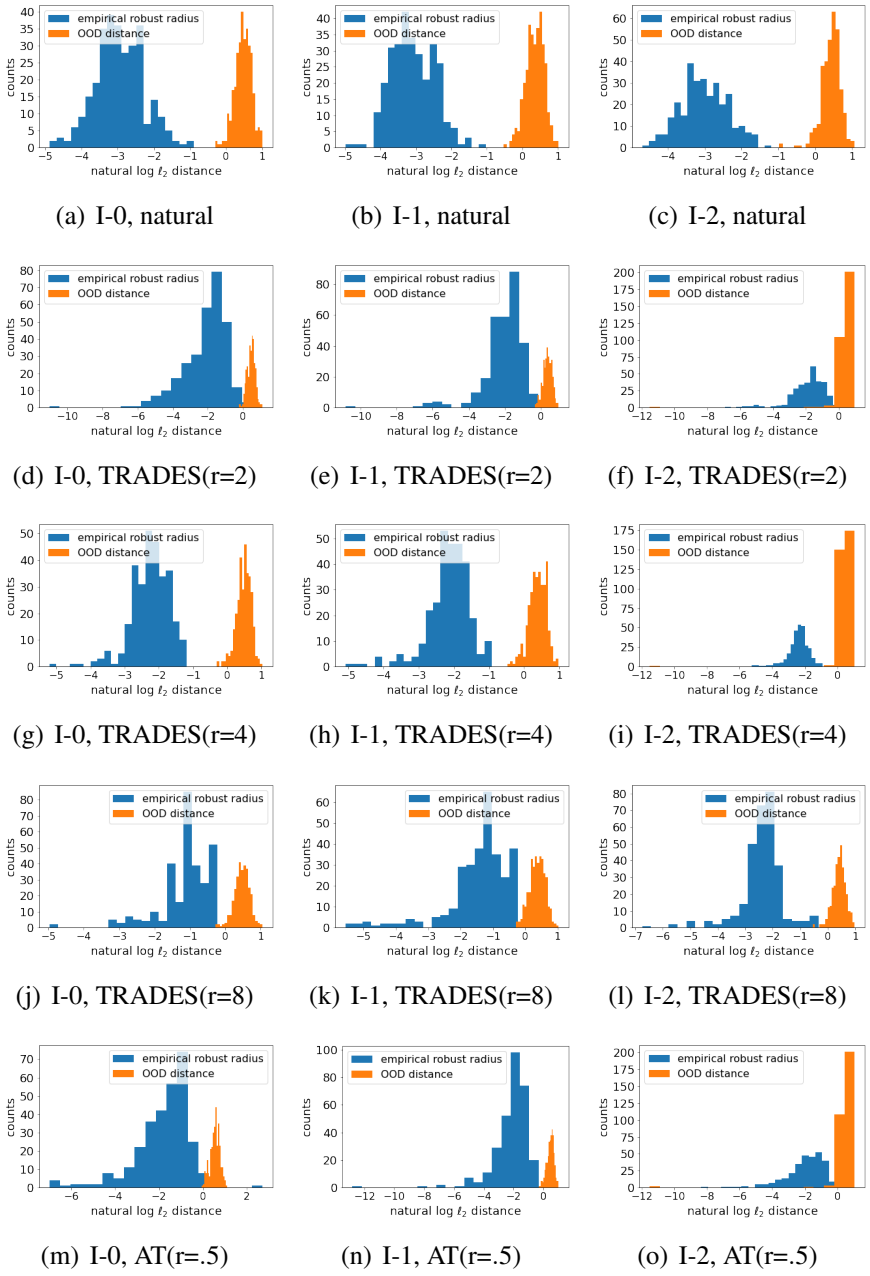


(q) C10-0, AT(r=1) (r) C10-4, AT(r=1) (s) C100-0, AT(r=1) (t) C100-4, AT(r=1)

Figure E.5: The histograms of the empirical robust radius and OOD distance for networks trained on CIFAR10-wo0 (C10-0), CIFAR10-wo4 (C10-4), CIFAR100-wo0 (C100-4), and CIFAR100-wo4 (C100-4) in the feature space.

E.2.5 Additional Results for the Corrupted Data

Robust models on corrupted data. On average (over the 90 and 75 corrupted sets), robust models have a NCG accuracy that is $1.35 \pm .02$, $1.36 \pm .03$, and $1.66 \pm .04$ times higher



(a) I-0, natural (b) I-1, natural (c) I-2, natural

(d) I-0, TRADES(r=2) (e) I-1, TRADES(r=2) (f) I-2, TRADES(r=2)

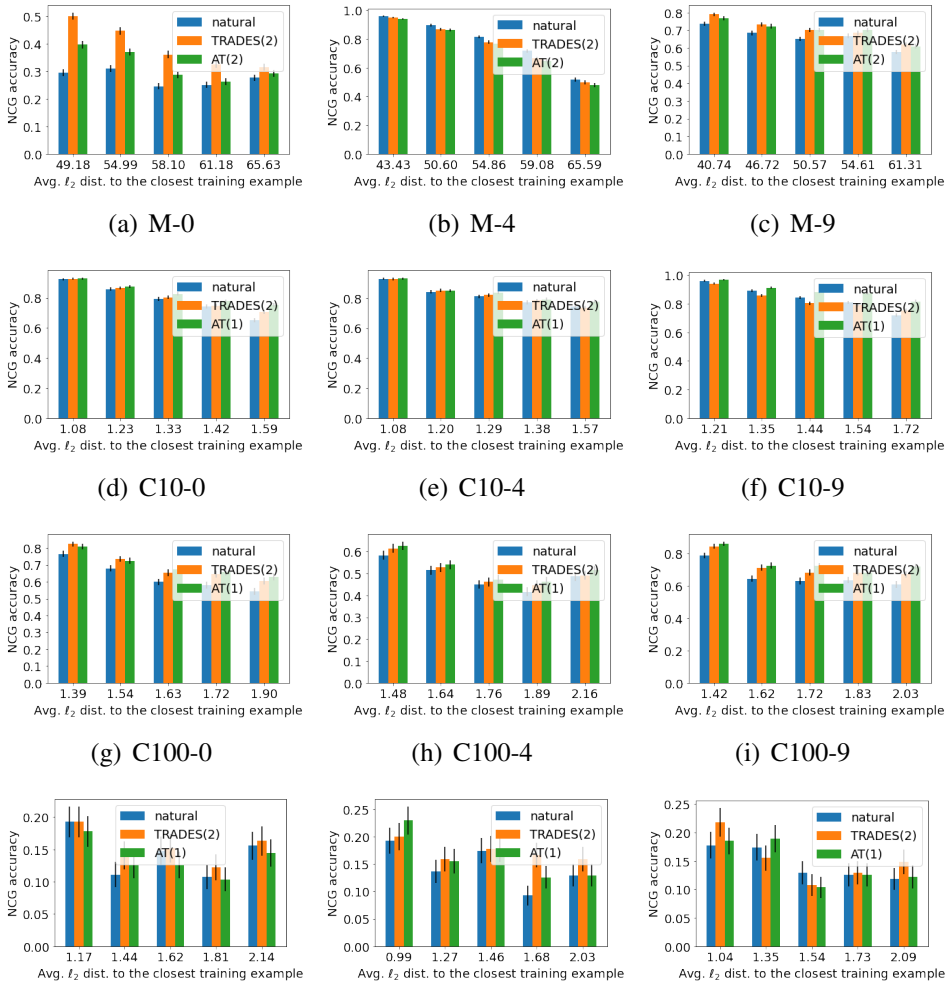
(g) I-0, TRADES(r=4) (h) I-1, TRADES(r=4) (i) I-2, TRADES(r=4)

(j) I-0, TRADES(r=8) (k) I-1, TRADES(r=8) (l) I-2, TRADES(r=8)

(m) I-0, AT(r=.5) (n) I-1, AT(r=.5) (o) I-2, AT(r=.5)

Figure E.6: The histograms of the empirical robust radius and OOD distance for networks trained on ImgNet100-wo0(I-0), ImgNet100-wo0(I-1), and ImgNet100-wo0(I-2) in the feature space.

than naturally trained models for CIFAR10, CIFAR100, and ImgNet100 respectively. In the feature space, we still find that **all** the 255 corruption sets have an NCG accuracy above chance level, but the NCG accuracies of the robust models are closer to the naturally trained models. For



(a) M-0 (b) M-4 (c) M-9

(d) C10-0 (e) C10-4 (f) C10-9

(g) C100-0 (h) C100-4 (i) C100-9

(j) I-0 (k) I-1 (l) I-2

Figure E.7: The NCG accuracy and the distance to the closest training example for MNIST, CIFAR10, CIFAR100, and ImageNet-100 in the pixel space.

CIFAR100, we still observe that **all** robust models have an NCG accuracy higher than the naturally trained models. But for CIFAR10, we find that on only 42 (out of 90) corrupted sets, TRADES(2) models have a higher NCG accuracy than naturally trained models. The average improvement over the naturally trained models in NCG accuracy goes down to $1.00 \pm .00$, $1.07 \pm .00$, and $1.09 \pm .01$ times for CIFAR10, CIFAR100, and ImgNet100 respectively.

E.2.6 Additional Results on the Slope of Corrupted Test Accuracy

NCG accuracy. Repeating the same experiment with NCG accuracy, we find similar results as well. In the pixel space, for CIFAR10 and CIFAR100, the slopes of the naturally trained models are significantly smaller than TRADES(2) on 15 and 14 (out of 18) corruption types. For ImgNet100, 6 out of 15 corruption types pass the test. The other 9 corruption types are not significant (they did not accept or reject the hypothesis). In the feature space, we also test whether the slopes of robust and naturally trained models are different. For CIFAR10 and CIFAR100, 17 and 15 (out of 18) corruption types, respectively, are not significantly different. For ImgNet100, 13 out of 15 corruption types are not significant. Figure E.9 shows the slope of the corrupted test and NCG accuracy for CIFAR10 and CIFAR100.

E.2.7 Full Table of Table 6.5

Table E.15 shows the full version of Table 6.5. We can derive the same conclusion from this table.

E.2.8 Most Predicted Classes

In Table E.16, we first remove a class from the training set of each dataset and train a neural network on the modified training set. We then look at the predictions of the neural network on these removed images and record their top two most predicted classes. From the result, we see that the outputs that these two networks produce follow some patterns no matter it is in the pixel space or the feature space, robust training or not For example, for C10-0 and C100-0, we see that "airplanes" are predicted as a ship or bird possibly because they have similar background of the sky. "aquatic mammals" are predicted as fish possibly because they are both in the water.

E.3 Proof of Sample Complexity Separation Theorem

As a warm-up, we prove Theorem 4.5.3 for $\mathbb{R}$ and $C = 2$ classes. We will use this as a building block for the general result.

E.3.1 Warm-up: Binary Case

For this special case, the universe for the examples will be the real line $\mathbb{R}$, and we consider a binary classification task with a third category that only appears in the testing distribution. Let $\varepsilon \in (0, 1/2)$ be a parameter. For the training distribution μ, we define four regions:

1. **Positive, large probability.** Let $\mathcal{P}_0 = [1, 2]$, labeled as "+".
2. **Positive, small probability.** Let $\mathcal{P}_1 = [3, 4]$, labeled as "+".
3. **Negative, large probability.** Let $\mathcal{N}_0 = [-2, -1]$, labeled as "−".
4. **Negative, small probability.** Let $\mathcal{N}_1 = [-4, -3]$, labeled as "−".

To sample from the training distribution μ, we first set $\ell \in \{-1, 1\}$ randomly with equal probability. Then, we choose $i \in \{0, 1\}$, where $i = 0$ with probability $1 - \varepsilon$ and $i = 1$ with probability ε. If $\ell = 1$, sample a point x uniformly from $\mathcal{P}_i$, and otherwise, if $\ell = -1$, we sample uniformly from $\mathcal{N}_i$. Note that with probability $1 - \varepsilon$, we have that $x \in \mathcal{P}_0 \cup \mathcal{N}_0$, while the probability of seeing any point in $\mathcal{P}_1 \cup \mathcal{N}_1$ is only ε. Finally, let ν be the uniform distribution on $[-6, -5] \cup [5, 6]$, where for $x \sim \nu$, we label it as $\mathsf{sign}(x)$.

We first argue that nearest category generalization can be efficiently solved. During training time, if we see at least 32 samples from μ, then with probability at least 99%, we will see samples from both $\mathcal{P}_0$ and $\mathcal{N}_0$, since $1 - \varepsilon > 1/2$, and we see samples from each class with equal probability. Therefore, once we have at least one sample from each class, we can construct the classifier decides ± 1 based on the midpoint of the training examples (which will be between -2

and $+2$ with good probability). Then, on the testing distribution ν, we see that all points will be classified correctly with the label of their nearest neighbor in the support of μ.

Turning to out-of-distribution detection, we claim that $\Omega(1/\varepsilon)$ samples are necessary. Indeed, to distinguish whether a sample comes from ν or from $\mathcal{P}_1 \cup \mathcal{N}_1$, we must see at least one sample from each of $\mathcal{P}_1$ and $\mathcal{N}_1$, since the support of ν is unknown at training time. As the probability of sampling from $\mathcal{P}_1$ or $\mathcal{N}_1$ is only ε, we will miss one of these regions with probability 99% if we have fewer than $t = 1/(100\varepsilon)$ samples from μ. Indeed, with probability $(1-\varepsilon)^t \geq e^{-\varepsilon t} = e^{-.01} > 0.99$, we have that all the samples come from $\mathcal{P}_0 \cup \mathcal{N}_0$.

E.3.2 General Case

We now provide the proof of Theorem 4.5.3 for any number $C \geq 2$ of classes and for any $d \geq 1$ dimensional dataset in $\mathbb{R}^d$ with nearest neighbors measured in ℓ_2 distance.

For $j \in \{1, 2, \ldots, C\}$ we define the following centers

$$a_0^j = 1 + 10j \qquad \text{and} \qquad a_1^j = 3 + 10j \qquad \text{and} \qquad a_2^j = 5 + 10j,$$

where we naturally embed them in d dimensions by using these as the value of the first coordinate and setting the rest of the coordinates to be zero. In other words, we define $\mathbf{a}_i^j = a_i^j \cdot \mathbf{e}_1$, where $\mathbf{e}_1$ is the standard basis vector, so that $\mathbf{a}_i^j \in \mathbb{R}^d$.

Then, for $i \in \{0, 1, 2\}$ and $j \in \{1, 2, \ldots, C\}$, we define the following regions, which are cubes centered at the points defined above and have side length $1/\sqrt{d}$. Formally, we consider the d-dimensional cubes

$$\mathcal{A}_i^j = \left\{ \mathbf{a}_i^j + (x_1, x_2, \ldots, x_d) \mid 0 \leq x_k \leq 1/\sqrt{d} \right\}.$$

To sample from the training distribution μ, we first choose $\ell \in \{1, 2, \ldots, C\}$ uniformly at

random. Then, we choose $i \in \{0,1\}$, where $i = 0$ with probability $1-\varepsilon$ and $i = 1$ with probability ε. Given our choice of ℓ, we sample a point $\mathbf{x}$ uniformly from $\mathcal{A}_i^\ell$. Note that with probability $1-\varepsilon$, we have that $\mathbf{x} \in \bigcup_{j=1}^C \mathcal{A}_0^j$, while the probability of seeing any point in $\bigcup_{j=1}^C \mathcal{A}_1^j$ is only ε. Finally, let ν be the uniform distribution on $\bigcup_{j=1}^C \mathcal{A}_2^j$. For both distributions, we label $\mathbf{x}$ as j if it comes from $\mathcal{A}_i^j$ for any $i \in \{0,1,2\}$.

Notice that this definition with $j = 0$ corresponds to the positively labeled regions $([1,2],[3,4],[5,6])$ from the proof of the binary case in the previous subsection. The probabilities are also the same when $C = 2$.

We explain the key properties of these regions, and then we prove the sample complexity results claimed in the theorem statement. First, for any $i \in \{0,1,2\}$ and $j \in \{1,2,\ldots,C\}$, if $\mathbf{x},\mathbf{y} \in \mathcal{A}_i^j$, then $\|\mathbf{x}-\mathbf{y}\|_2 \leq 1$ because each $\mathcal{A}_i^j$ is a cube with side length $1/\sqrt{d}$ in $\mathbb{R}^d$.

Next, consider $\mathbf{x} \in \mathcal{A}_2^j$. We claim that $\mathbf{x}$ is closer to $\mathcal{A}_0^j$ than to any point $\mathbf{z} \in \mathcal{A}_0^{j'} \cup \mathcal{A}_1^{j'}$ for any $j' \neq j$. To see this, we can check that the triangle inequality implies that

$$\min_{\mathbf{y} \in \mathcal{A}_0^j} \|\mathbf{x}-\mathbf{y}\|_2 \leq 4+1 = 5,$$

while, since the centers satisfy $|a_2^j - a_1^{j'}| > |a_2^j - a_0^{j'}| \geq 6$, we also have that for $j' \neq j$,

$$\min_{\mathbf{z} \in \mathcal{A}_0^{j'} \cup \mathcal{A}_1^{j'}} \|\mathbf{x}-\mathbf{z}\|_2 \geq 6.$$

As a consequence, we have that the nearest neighbor in ℓ_2 distance for any point $\mathbf{x} \in \mathcal{A}_2^j$ has the same label j as $\mathbf{x}$ does. In particular, this implies that we can solve the nearest category generalization problem for points sampled from ν. To do so, we first sample $\Theta(C\log C)$ points from μ, so that by a coupon collector argument, we see at least one point from $\mathcal{A}_0^j$ for each $j \in \{1,2,\ldots,C\}$. Then, recall that ν is supported on the union of $\mathcal{A}_2^{j'}$ over $j' \in \{1,2,\ldots,C\}$. By the above calculations, we have that the nearest neighbor for a point $\mathbf{x} \in \mathcal{A}_2^j$ is some point from

either $\mathcal{A}_0^j$ or $\mathcal{A}_1^j$. Therefore, since we have sampled at least one point from $\mathcal{A}_0^j$, we can correctly determine that $\mathbf{x}$ has label j by computing the nearest neighbor in our sampled points. To be more precise, we can compute the multi-class large-margin classifier, where we have sequential decision regions (corresponding roughly to the centers defined above), setting the decision boundaries to be equally spaced between samples from adjacent regions (i.e., the natural generalization of the 1D large-margin solution). Importantly, this solution does not require any extra knowledge of the support of μ and ν because it can be computed directly from the samples (and we have argued that with $\Theta(C \log C)$ samples, we will see all C classes at least once).

We turn our attention to our lower bound, which is that we need at least $\Omega(C/\varepsilon)$ samples to solve the OOD detection problem. More precisely, we provide a lower bound for the number of samples to guarantee that we see that least one point from each region $\mathcal{A}_1^j$ for each $j \in \{1, 2, \ldots, C\}$. This is a prerequisite for solving the OOD detection problem, because otherwise, we cannot tell whether a point comes from μ or ν without prior knowledge of the regions. For the lower bound, we use the same argument as in the binary case in the previous subsection. This implies that we need $\Omega(C/\varepsilon)$ samples to see one from $\mathcal{A}_1^j$ for each fixed j since the probability of sampling from this region is ε/C by the definition of μ.

E.3.3 Alternative Generalizations

We could also use a "noisy one-hot encoding" to prove the theorem, replicating and rotating the 1D dataset $\log_2 C$ times, to get a subset of $\mathbb{R}^{\log_2 C}$ for C classes. One dimension is non-zero for each point, and each dimension has points from two possible labels (C total labels). Use $6C$ regions to define the low probability, high probability, and OOD regions (6 in each dimension with 3 for each class). Again, by a coupon collector argument, we will see some point from each of the high probability regions after $O(C \log C)$ samples. This enables nearest category generalization. On the other hand, for OOD detection, we need $\Omega(C/\varepsilon)$ samples, where ε is the sample probability from a low probability region.

Instead of boxes, we could use Gaussian distributions with covariance $\sigma^2 I_d$ and means shifted by increments of a vector, spacing out the means by distance $\Omega(\sigma\sqrt{\log(d/\epsilon)})$ to get analogous guarantees. Similar ideas work for Hamming distance on $\{0,1\}^d$; embed regions as intervals in the partial order along a path from 0^d to 1^d, spacing them out to ensure 1-NN properties. In general, there are many metric spaces where we can provide a separation between nearest category generalization and OOD detection by correctly setting up the regions and sampling probabilities. Therefore, we believe it a general phenomena that nearest category generalization is a more tractable goal, in terms of sample complexity, than OOD detection.

Table E.11: The average empirical robust radius, average OOD distance, percentage of OOD examples covered by the robust norm ball of its closest training example and the NCG accuracy (in the pixel space of MNIST datasets).

		empirical robust radius	OOD dist.	portion covered	NCG acc.
M-0	AT(2)	2.33	7.00	0.00	0.46
	TRADES(2)	2.17	6.98	0.00	0.46
	TRADES(4)	2.07	6.94	0.00	0.48
	TRADES(8)	0.68	6.97	0.00	0.42
	natural	1.26	6.95	0.00	0.36
M-1	AT(2)	1.83	4.30	0.00	0.51
	TRADES(2)	1.57	4.31	0.00	0.35
	TRADES(4)	1.44	4.31	0.00	0.54
	TRADES(8)	0.56	4.29	0.00	0.28
	natural	0.89	4.33	0.00	0.23
M-2	AT(2)	2.27	6.92	0.00	0.54
	TRADES(2)	2.15	6.93	0.00	0.53
	TRADES(4)	2.23	6.86	0.00	0.52
	TRADES(8)	0.71	7.03	0.00	0.47
	natural	1.33	6.98	0.00	0.40
M-3	AT(2)	2.35	6.27	0.00	0.66
	TRADES(2)	2.22	6.33	0.00	0.66
	TRADES(4)	1.93	6.30	0.00	0.58
	TRADES(8)	0.70	6.20	0.01	0.55
	natural	1.52	6.34	0.00	0.59
M-4	AT(2)	2.34	5.64	0.00	0.74
	TRADES(2)	2.32	5.89	0.00	0.81
	TRADES(4)	1.84	5.65	0.00	0.76
	TRADES(8)	0.66	5.83	0.00	0.75
	natural	1.33	5.73	0.00	0.76
M-5	AT(2)	2.39	6.30	0.00	0.61
	TRADES(2)	2.17	6.34	0.00	0.62
	TRADES(4)	2.24	6.37	0.00	0.62
	TRADES(8)	0.66	6.33	0.00	0.55
	natural	1.39	6.34	0.00	0.53
M-6	AT(2)	2.41	6.46	0.00	0.55
	TRADES(2)	2.18	6.45	0.00	0.56
	TRADES(4)	2.06	6.52	0.00	0.49
	TRADES(8)	0.58	6.46	0.00	0.53
	natural	1.30	6.44	0.00	0.51
M-7	AT(2)	2.18	5.55	0.00	0.67
	TRADES(2)	2.10	5.47	0.00	0.72
	TRADES(4)	1.88	5.51	0.01	0.72
	TRADES(8)	0.77	5.53	0.01	0.59
	natural	1.27	5.51	0.00	0.53
M-8	AT(2)	2.22	6.32	0.00	0.49
	TRADES(2)	1.99	6.30	0.00	0.51
	TRADES(4)	1.92	6.35	0.00	0.47
	TRADES(8)	0.63	6.30	0.00	0.45
	natural	1.35	6.30	0.00	0.42
M-9	AT(2)	2.33	5.14	0.00	0.71
	TRADES(2)	2.28	5.25	0.00	0.68
	TRADES(4)	2.18	5.13	0.00	0.70
	TRADES(8)	0.66	5.20	0.00	0.65
	natural	1.45	5.08	0.00	0.58

Table E.12: The average empirical robust radius, average OOD distance, percentage of OOD examples covered by the robust norm ball of its closest training example and the NCG accuracy (in the pixel space of C10, C100, and I).

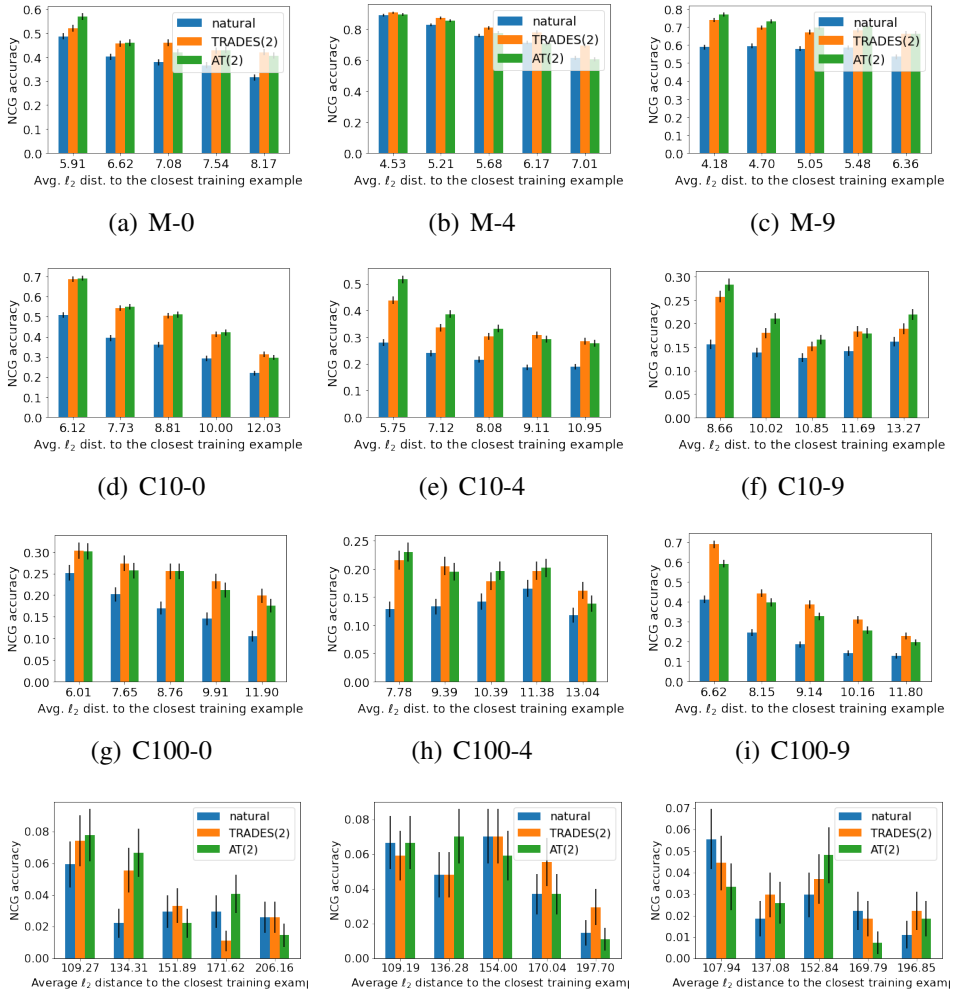
		empirical robust radius	OOD dist.	portion covered	NCG acc.
C10-0	AT(2)	2.14	8.67	0.00	0.49
	TRADES(2)	2.17	8.89	0.00	0.49
	TRADES(4)	1.62	9.13	0.00	0.52
	TRADES(8)	0.51	8.67	0.00	0.48
	natural	0.09	8.71	0.00	0.35
C10-4	AT(2)	1.92	8.04	0.00	0.36
	TRADES(2)	1.75	8.29	0.00	0.33
	TRADES(4)	0.93	8.30	0.00	0.33
	TRADES(8)	0.87	8.16	0.00	0.29
	natural	0.09	8.29	0.00	0.22
C10-9	AT(2)	2.11	10.80	0.00	0.21
	TRADES(2)	2.01	11.05	0.00	0.19
	TRADES(4)	0.92	10.83	0.00	0.25
	TRADES(8)	0.65	10.83	0.00	0.25
	natural	0.12	10.85	0.00	0.14
C100-0	AT(2)	1.96	8.89	0.00	0.24
	TRADES(2)	1.93	9.03	0.00	0.25
	TRADES(4)	0.82	9.01	0.00	0.25
	TRADES(8)	0.59	9.10	0.00	0.21
	natural	0.11	8.94	0.00	0.17
C100-4	AT(2)	2.01	10.43	0.00	0.19
	TRADES(2)	2.21	10.53	0.00	0.19
	TRADES(4)	1.12	10.57	0.00	0.19
	TRADES(8)	0.41	10.36	0.00	0.18
	natural	0.10	10.17	0.00	0.14
C100-9	AT(2)	2.10	9.18	0.00	0.35
	TRADES(2)	2.18	9.57	0.00	0.41
	TRADES(4)	0.86	9.27	0.00	0.43
	TRADES(8)	1.49	9.01	0.00	0.47
	natural	0.08	9.22	0.00	0.22
I-0	AT(2)	2.77	155.07	0.00	0.04
	TRADES(2)	3.14	155.62	0.00	0.04
	TRADES(4)	3.54	160.05	0.00	0.06
	TRADES(8)	2.67	161.10	0.00	0.07
	natural	0.29	157.42	0.00	0.03
I-1	AT(2)	2.72	153.03	0.00	0.05
	TRADES(2)	2.85	155.91	0.00	0.05
	TRADES(4)	2.92	156.97	0.00	0.06
	TRADES(8)	2.32	157.91	0.00	0.07
	natural	0.26	152.47	0.00	0.05
I-2	AT(2)	2.22	154.28	0.00	0.03
	TRADES(2)	2.49	155.76	0.00	0.03
	TRADES(4)	2.67	156.83	0.00	0.04
	TRADES(8)	2.27	156.27	0.00	0.05
	natural	0.15	154.16	0.00	0.03

Table E.13: The average empirical robust radius, average OOD distance, percentage of OOD examples covered by the robust norm ball of its closest training example and the NCG accuracy (in the feature space of MNIST datasets).

		empirical robust radius	OOD dist.	portion covered	NCG acc.
M-0	AT(2)	7.50	57.78	0.00	0.32
	TRADES(2)	9.08	57.16	0.00	0.39
	TRADES(4)	12.54	57.28	0.00	0.49
	TRADES(8)	15.91	57.50	0.00	0.55
	natural	5.84	57.78	0.00	0.28
M-1	AT(2)	6.12	37.06	0.00	0.20
	TRADES(2)	7.69	37.02	0.00	0.26
	TRADES(4)	10.51	37.01	0.00	0.51
	TRADES(8)	13.68	36.87	0.00	0.50
	natural	4.80	37.06	0.00	0.13
M-2	AT(2)	13.13	64.51	0.00	0.46
	TRADES(2)	10.77	63.96	0.00	0.53
	TRADES(4)	14.25	62.53	0.00	0.59
	TRADES(8)	17.60	64.77	0.00	0.62
	natural	7.25	62.41	0.00	0.41
M-3	AT(2)	15.55	70.11	0.00	0.71
	TRADES(2)	13.34	70.09	0.00	0.73
	TRADES(4)	17.64	70.68	0.00	0.73
	TRADES(8)	21.42	69.82	0.00	0.74
	natural	9.33	70.11	0.00	0.68
M-4	AT(2)	10.97	54.27	0.00	0.73
	TRADES(2)	13.43	53.89	0.00	0.77
	TRADES(4)	16.79	54.23	0.00	0.81
	TRADES(8)	20.62	53.80	0.00	0.86
	natural	9.74	54.27	0.00	0.78
M-5	AT(2)	14.92	65.51	0.00	0.63
	TRADES(2)	12.25	65.25	0.00	0.65
	TRADES(4)	15.50	64.37	0.00	0.68
	TRADES(8)	19.64	65.12	0.00	0.69
	natural	9.47	65.51	0.00	0.61
M-6	AT(2)	11.66	60.66	0.00	0.58
	TRADES(2)	10.62	60.65	0.00	0.60
	TRADES(4)	14.15	60.67	0.00	0.65
	TRADES(8)	17.44	60.28	0.00	0.66
	natural	7.42	60.66	0.00	0.54
M-7	AT(2)	12.03	51.40	0.00	0.54
	TRADES(2)	10.80	52.75	0.00	0.61
	TRADES(4)	14.09	51.78	0.00	0.68
	TRADES(8)	19.22	52.96	0.00	0.67
	natural	7.49	51.40	0.00	0.53
M-8	AT(2)	11.88	60.31	0.00	0.47
	TRADES(2)	10.88	60.31	0.00	0.51
	TRADES(4)	15.79	61.43	0.00	0.56
	TRADES(8)	17.15	59.64	0.00	0.59
	natural	7.43	60.31	0.00	0.46
M-9	AT(2)	11.28	51.54	0.00	0.71
	TRADES(2)	13.13	52.28	0.00	0.71
	TRADES(4)	16.99	50.70	0.00	0.74
	TRADES(8)	20.84	51.48	0.00	0.80
	natural	8.12	52.32	0.00	0.61

Table E.14: The average empirical robust radius, average OOD distance, percentage of OOD examples covered by the robust norm ball of its closest training example and the NCG accuracy (in the feature space of C10, C100, and I).

		empirical robust radius	OOD dist.	portion covered	NCG acc.
C10-0	AT(1)	0.65	1.33	0.01	0.83
	TRADES(2)	0.62	1.33	0.01	0.81
	TRADES(4)	0.27	1.31	0.00	0.83
	TRADES(8)	0.43	1.31	0.00	0.83
	natural	0.48	1.31	0.00	0.80
C10-4	AT(1)	0.69	1.31	0.01	0.84
	TRADES(2)	0.68	1.29	0.01	0.82
	TRADES(4)	0.28	1.31	0.00	0.85
	TRADES(8)	0.45	1.31	0.00	0.85
	natural	0.57	1.31	0.00	0.82
C10-9	AT(1)	0.93	1.43	0.07	0.89
	TRADES(2)	0.85	1.46	0.06	0.83
	TRADES(4)	0.30	1.43	0.00	0.88
	TRADES(8)	0.60	1.43	0.00	0.87
	natural	0.73	1.43	0.03	0.84
C100-0	AT(1)	0.51	1.64	0.00	0.70
	TRADES(2)	0.45	1.64	0.00	0.69
	TRADES(4)	0.29	1.65	0.00	0.68
	TRADES(8)	0.35	1.65	0.00	0.68
	natural	0.30	1.65	0.00	0.63
C100-4	AT(1)	0.64	1.84	0.00	0.74
	TRADES(2)	0.55	1.83	0.00	0.75
	TRADES(4)	0.31	1.83	0.00	0.73
	TRADES(8)	0.45	1.83	0.00	0.74
	natural	0.36	1.83	0.00	0.69
C100-9	AT(1)	0.63	1.75	0.02	0.74
	TRADES(2)	0.52	1.75	0.00	0.72
	TRADES(4)	0.36	1.73	0.00	0.71
	TRADES(8)	0.45	1.73	0.00	0.71
	natural	0.33	1.73	0.00	0.66
I-0	AT(.5)	0.32	1.78	0.00	0.16
	TRADES(2)	0.20	1.63	0.00	0.15
	TRADES(4)	0.12	1.66	0.00	0.12
	TRADES(8)	0.39	1.64	0.00	0.13
	natural	0.07	1.64	0.00	0.11
I-1	AT(.5)	0.23	1.65	0.00	0.15
	TRADES(2)	0.18	1.50	0.00	0.18
	TRADES(4)	0.14	1.51	0.00	0.14
	TRADES(8)	0.34	1.49	0.00	0.15
	natural	0.05	1.47	0.00	0.13
I-2	AT(.5)	0.21	1.67	0.01	0.15
	TRADES(2)	0.22	1.57	0.00	0.15
	TRADES(4)	0.11	1.57	0.00	0.14
	TRADES(8)	0.12	1.56	0.00	0.14
	natural	0.06	1.58	0.00	0.11



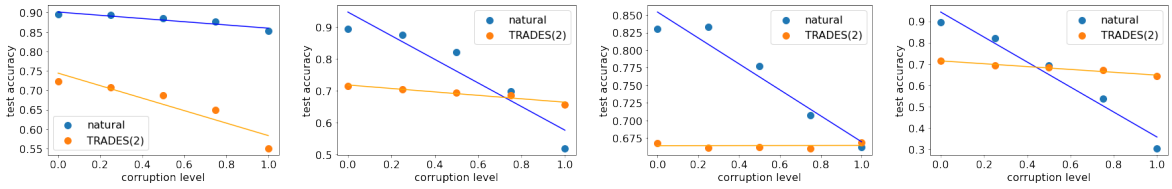
(a) M-0 (b) M-4 (c) M-9

(d) C10-0 (e) C10-4 (f) C10-9

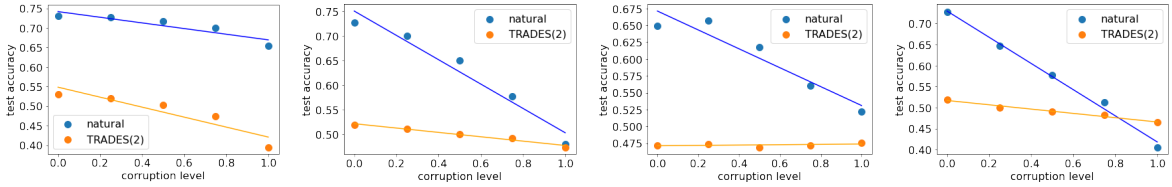
(g) C100-0 (h) C100-4 (i) C100-9

(j) I-0 (k) I-1 (l) I-2

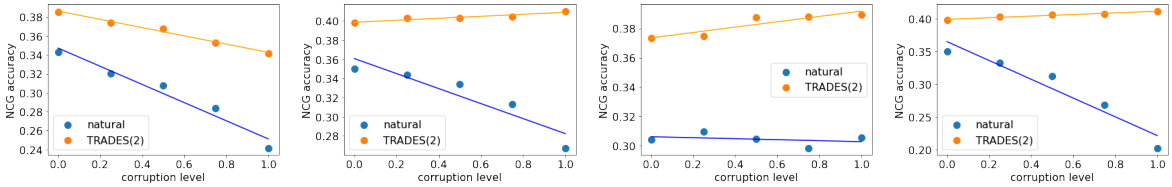
Figure E.8: The NCG accuracy and the distance to the closest training example for MNIST, CIFAR10, CIFAR100, and ImageNet-100 in the feature space.



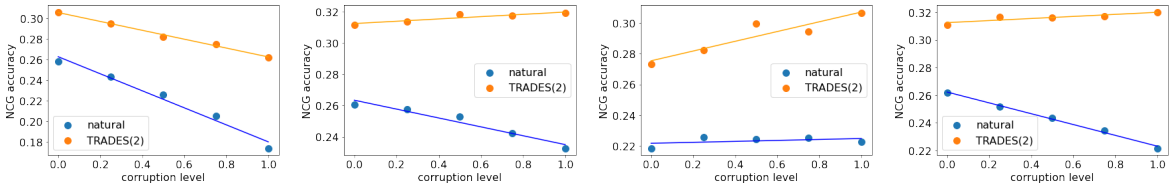
(a) C10 brightness (b) C10 defocus (c) C10 elastic transform (d) C10 Gaussian blur



(e) C100 brightness (f) C100 defocus (g) C100 elastic transform (h) C100 Gaussian blur



(i) C10 brightness (j) C10 defocus (k) C10 elastic transform (l) C10 Gaussian blur



(m) C100 brightness (n) C100 defocus (o) C100 elastic transform (p) C100 Gaussian blur

Figure E.9: The slopes of the test and NCG accuracies of naturally trained models and TRADES(2) on CIFAR10 and CIFAR100 in the pixel space.

Table E.15: We show four different metrics on models trained on CIFAR10 and CIFAR100 and evaluated on the Gaussian noise corrupted data. We show the NCG accuracy, test accuracy, the test accuracy on the NCG correct corrupted examples, the test accuracy on the NCG incorrect corrupted examples, and the distance to the closest training example.

	model		natural				TRADES(2)			
	dataset	level	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.	tst acc.	NCG incorrect tst acc.	NCG correct tst acc.	NCG acc.
pixel	C10	1	0.76	0.70	0.88	0.34	0.71	0.67	0.78	0.40
		2	0.63	0.54	0.82	0.30	0.71	0.66	0.78	0.39
		3	0.48	0.39	0.75	0.26	0.70	0.65	0.77	0.39
		4	0.41	0.32	0.70	0.24	0.69	0.63	0.77	0.38
		5	0.36	0.27	0.66	0.22	0.68	0.63	0.77	0.38
	C100	1	0.63	0.56	0.84	0.25	0.52	0.43	0.72	0.30
		2	0.55	0.47	0.79	0.24	0.51	0.43	0.71	0.30
		3	0.47	0.39	0.74	0.23	0.51	0.42	0.71	0.30
		4	0.44	0.36	0.71	0.22	0.50	0.42	0.71	0.30
		5	0.40	0.33	0.67	0.21	0.50	0.41	0.71	0.29
	I	1	0.42	0.41	0.68	0.04	0.36	0.35	0.51	0.06
		2	0.34	0.33	0.64	0.03	0.36	0.35	0.53	0.05
		3	0.22	0.21	0.49	0.03	0.34	0.33	0.49	0.05
		4	0.12	0.11	0.24	0.02	0.30	0.30	0.45	0.05
		5	0.04	0.04	0.07	0.02	0.22	0.22	0.34	0.04
feature	C10	1	0.74	0.39	0.78	0.89	0.72	0.32	0.77	0.89
		2	0.59	0.35	0.64	0.85	0.56	0.23	0.62	0.85
		3	0.45	0.33	0.48	0.82	0.40	0.19	0.45	0.83
		4	0.39	0.33	0.40	0.81	0.35	0.20	0.38	0.83
		5	0.34	0.28	0.35	0.82	0.31	0.18	0.33	0.83
	C100	1	0.60	0.25	0.72	0.74	0.62	0.29	0.71	0.78
		2	0.51	0.24	0.63	0.68	0.53	0.29	0.62	0.74
		3	0.43	0.23	0.54	0.64	0.44	0.25	0.53	0.69
		4	0.40	0.22	0.51	0.63	0.40	0.23	0.49	0.67
		5	0.37	0.21	0.46	0.61	0.37	0.21	0.46	0.65
	I	1	0.22	0.18	0.44	0.15	0.21	0.18	0.41	0.16
		2	0.19	0.16	0.36	0.14	0.18	0.15	0.34	0.15
		3	0.14	0.12	0.26	0.14	0.13	0.11	0.21	0.17
		4	0.09	0.08	0.16	0.13	0.08	0.07	0.14	0.16
		5	0.05	0.04	0.08	0.14	0.04	0.03	0.08	0.14

Table E.16: This table shows the top and second most predicted classes on the examples of unseen classes for each dataset.

			unseen class	top most predicted class	second most predicted class
M-0	pixel	natural	0	6	2
		TRADES(2)	0	2	6
	feature	natural	0	6	2
		TRADES(2)	0	6	2
M-4	pixel	natural	4	9	7
		TRADES(2)	4	9	7
	feature	natural	4	9	7
		TRADES(2)	4	9	7
M-9	pixel	natural	9	4	7
		TRADES(2)	9	4	7
	feature	natural	9	4	8
		TRADES(2)	9	4	8
C100-0	pixel	natural	aquatic mammals	fish	small mammals
		TRADES(2)	aquatic mammals	fish	medium-sized mammals
	feature	natural	aquatic mammals	fish	reptiles
		TRADES(2)	aquatic mammals	fish	reptiles
C100-4	pixel	natural	fruit and vegetables	flowers	food containers
		TRADES(2)	fruit and vegetables	flowers	food containers
	feature	natural	fruit and vegetables	flowers	food containers
		TRADES(2)	fruit and vegetables	flowers	food containers
C100-9	pixel	natural	large man-made outdoor things	large natural outdoor scenes	vehicles 2
		TRADES(2)	large man-made outdoor things	large natural outdoor scenes	trees
	feature	natural	large man-made outdoor things	large natural outdoor scenes	vehicles 2
		TRADES(2)	large man-made outdoor things	large natural outdoor scenes	vehicles 2
C10-0	pixel	natural	airplane	ship	bird
		TRADES(2)	airplane	ship	bird
	feature	natural	airplane	ship	bird
		TRADES(2)	airplane	ship	bird
C10-4	pixel	natural	deer	bird	horse
		TRADES(2)	deer	frog	bird
	feature	natural	deer	bird	horse
		TRADES(2)	deer	bird	cat
C10-9	pixel	natural	truck	automobile	airplane
		TRADES(2)	truck	automobile	ship
	feature	natural	truck	automobile	ship
		TRADES(2)	truck	automobile	ship
I-0	pixel	natural	American robin	lorikeet	stinkhorn mushroom
		TRADES(2)	American robin	lorikeet	hare
	feature	natural	American robin	hare	little blue heron
		TRADES(2)	American robin	hare	little blue heron
I-1	pixel	natural	Gila monster	eastern hog-nosed snake	dung beetle
		TRADES(2)	Gila monster	eastern hog-nosed snake	rock crab
	feature	natural	Gila monster	eastern hog-nosed snake	dung beetle
		TRADES(2)	Gila monster	eastern hog-nosed snake	dung beetle
I-2	pixel	natural	eastern hog-nosed snake	garter snake	Gila monster
		TRADES(2)	eastern hog-nosed snake	garter snake	Gila monster
	feature	natural	eastern hog-nosed snake	Gila monster	garter snake
		TRADES(2)	eastern hog-nosed snake	Gila monster	dung beetle

Appendix F

Understanding Rare Spurious Correlations in Neural Networks

F.1 Experimental Details and Additional Results

F.1.1 Detailed Experimental Setups

For MNIST and Fashion, we set the learning rate to 0.01 for all architectures and optimizers. We set the momentum to 0.9 when the optimizer is SGD. For CIFAR10, when training with SGD, we set the learning rate to 0.1 for ResNets trained and 0.01 for Vgg16 because Vgg16 failed to converge with a learning rate of 0.1. We set the learning rate to 0.01 for ResNets when running with Adam (Vgg16 failed to converge with Adam). We use a learning rate scheduler that decreases the learning rate by a factor of 0.1 on the 40-th, 50-th, and 60-th epoch. The code for all the experiments is available at `https://github.com/yangarbiter/rare-spurious-correlation`.

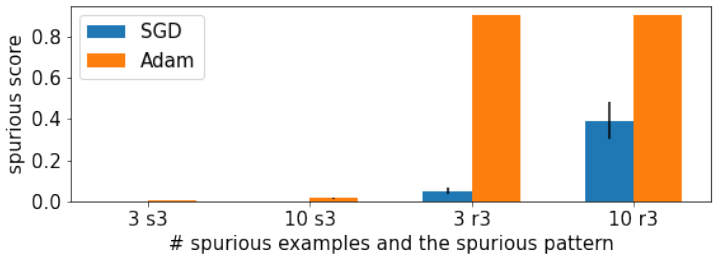
F.1.2 How the Optimization Process Effect Spurious Scores

We study how the optimization process affects the learning of the spurious correlation. We look into two main components of the optimization process, the optimizers and the use of gradient clipping, and examine how each component affects the spurious score. For the optimizers, we compare between Adam and SGD, while for gradient clipping, we compare between no gradient clipping and clipping the norm the gradient to 0.1. We repeat the five times with different random seeds and record their mean and standard error.

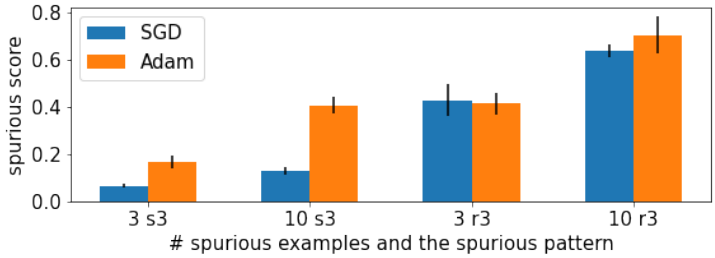
Figure F.1 shows the average spurious scores between different optimizers on different datasets and spurious patterns. We see that the networks trained with Adam, in general, have larger or even average spurious scores than the networks trained with SGD. This result indicates that Adam can be more susceptible to learning spurious correlations than SGD.

Figure F.2 shows the average spurious scores on networks trained with and without gradient clipping on different datasets and spurious patterns. We see that, in most cases, with and without gradient clipping perform similarly. It appears that using gradient clipping alone is not sufficient to eliminate the rare spurious correlations from neural networks.

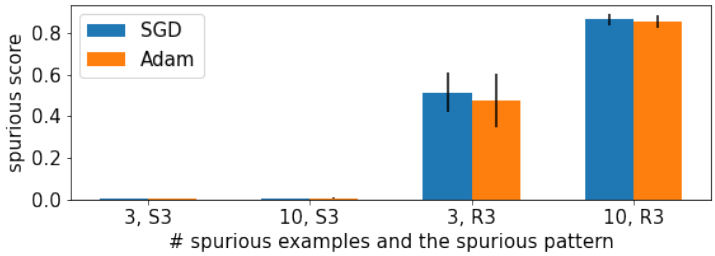
Overall, we see that rare spurious correlations are learned regardless of the choice of the optimizer and whether the gradient clipping is performed or not. This indicates that tweaking individual components in the optimization process may not be sufficient to remove spurious correlations.



(a) MNIST, SGD vs. Adam

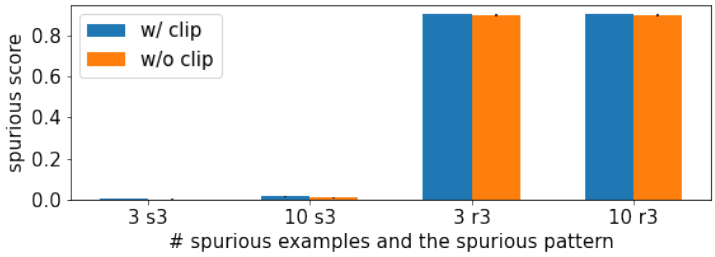


(b) Fashion, SGD vs. Adam

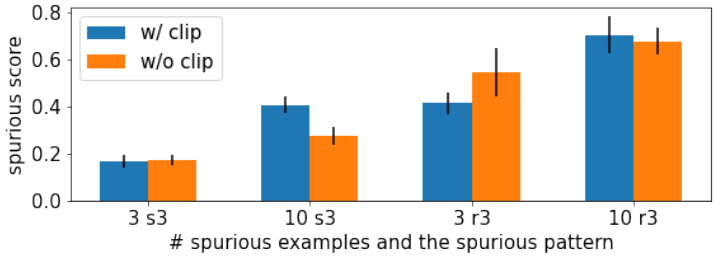


(c) CIFAR10, SGD vs. Adam

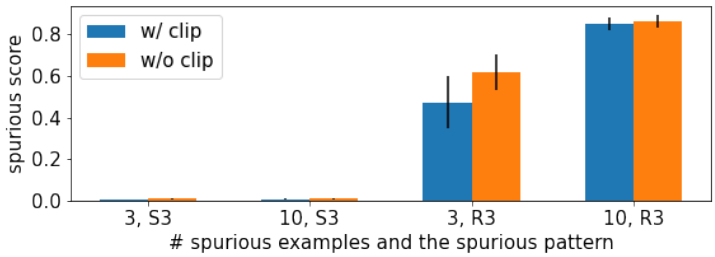
Figure F.1: The mean and standard error of the spurious scores on neural networks trained with SGD versus Adam. We consider networks trained with three and ten spurious examples as well as using the *S3* and *R3* patterns.



(a) MNIST, w/ vs. w/o clipping



(b) Fashion, w/ vs. w/o clipping



(c) CIFAR10, w/ vs. w/o clipping

Figure F.2: The mean and standard error of the spurious scores on neural networks trained with and without gradient clipping. We consider networks trained with three and ten spurious examples as well as using the *S3* and *R3* patterns.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Conference on Computer and Communications Security*, pages 308–318, 2016.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, volume 31, pages 9505–9515, 2018.

[3] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems*, pages 12192–12202, 2019.

[4] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[5] Laurent Amsaleg, James Bailey, Dominique Barbe, Sarah Erfani, Michael E Houle, Vinh Nguyen, and Miloš Radovanović. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *IEEE Workshop on Information Forensics and Security*, pages 1–6, 2017.

[6] Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. *arXiv preprint arXiv:1906.03526*, 2019.

[7] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, 2019.

[8] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[9] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, volume 70, pages 233–242, 2017.

[10] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018.

[11] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[12] Joseph L Austerweil, Shi Xian Liew, Nolan Conaway, and Kenneth J Kurtz. Creating something different: Similarity, contrast, and representativeness in categorization. 2019.

[13] Pranjal Awasthi, Abhratanu Dutta, and Aravindan Vijayaraghavan. On robustness to adversarial examples and polynomial optimization. In *Advances in Neural Information Processing Systems*, pages 13737–13747, 2019.

[14] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, pages 528–539, 2020.

[15] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[16] Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pages 715–724, 2020.

[17] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275, 2017.

[18] Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017.

[19] Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[20] Dimitris Bertsimas, Arthur Delarue, Patrick Jaillet, and Sebastien Martin. The price of interpretability. *arXiv preprint arXiv:1907.03419*, 2019.

[21] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower bounds on adversarial robustness from optimal transport. In *Advances in Neural Information Processing Systems*, pages 7496–7508, 2019.

[22] Robi Bhattacharjee and Kamalika Chaudhuri. When are non-parametric methods robust? *arXiv preprint arXiv:2003.06121*, 2020.

[23] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 387–402, 2013.

[24] Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: rigorous

guarantees and inherent limitations. *arXiv preprint arXiv:1911.07375*, 2019.

[25] Guy Blanc, Jane Lange, and Li-Yang Tan. Provable guarantees for decision tree induction: the agnostic setting. *arXiv preprint arXiv:2006.00743*, 2020.

[26] Naama Boer, Daniel Deutch, Nave Frost, and Tova Milo. Personal insights for altering decisions of tree-based ensembles over time. *Proceedings of the VLDB Endowment*, 13(6): 798–811, 2020.

[27] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207, 2003.

[28] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[30] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[31] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[32] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[33] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *arXiv preprint arXiv:1907.01003*, 2019.

[34] Alon Brutzkus, Amit Daniely, and Eran Malach. On the optimality of trees generated by id3. *arXiv preprint arXiv:1907.05444*, 2019.

[35] Alon Brutzkus, Amit Daniely, and Eran Malach. Id3 learns juntas for smoothed product distributions. In *Conference on Learning Theory*, pages 902–915, 2020.

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

[37] Cody Burkard and Brent Lagesse. Analysis of causative attacks against svms learning from data streams. In *International Workshop on Security and Privacy Analytics*, pages 31–36, 2017.

[38] Nicholas Carlini. *Evaluation and Design of Robust Neural Network Defenses*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2018.

[39] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks.

In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[40] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. On Evaluating Adversarial Robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[41] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, pages 267–284, 2019.

[42] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

[43] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11190–11201, 2019.

[44] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology*, 2(3):27, 2011.

[45] Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 3437–3445, 2014.

[46] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[47] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust Decision Trees Against Adversarial Examples. In *International Conference on Machine Learning*, 2019.

[48] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. *arXiv preprint arXiv:2003.12862*, 2020.

[49] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[50] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient Hard-label Black-box Attack: An Optimization-based Approach. In *International Concerence on Learning Representations*, 2019.

[51] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. *arXiv preprint arXiv:1909.10773*, 2019.

[52] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.

[53] Benjamin Cosman, Madeline Endres, Georgios Sakkas, Leon Medvinsky, Yao-Yuan Yang, Ranjit Jhala, Kamalika Chaudhuri, and Westley Weimer. Pablo: Helping novices debug python code through data-driven fault localization. In *ACM Technical Symposium on Computer Science Education*, pages 1047–1053, 2020.

[54] Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[55] 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*, 2019.

[56] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.

[57] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[58] Daniel Deutch and Nave Frost. Constraints-based explanations of classifications. In *IEEE International Conference on Data Engineering*, pages 530–541, 2019.

[59] Luc Devroye, Laszlo Gyorfi, Adam Krzyzak, and Gabor Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, pages 1371–1385, 1994.

[60] Elvis Dohmatob. Generalized no free lunch theorem for adversarial robustness. In *International Conference on Machine Learning*, pages 1646–1654, 2019.

[61] Xin Dong, Yaxin Zhu, Yupeng Zhang, Zuohui Fu, Dongkuan Xu, Sen Yang, and Gerard De Melo. Leveraging adversarial training in self-learning for cross-lingual text classification. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1541–1544, 2020.

[62] Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM*, 61(1):1, 2014.

[63] Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. Defense against adversarial images using web-scale nearest-neighbor search. *arXiv preprint arXiv:1903.01612*, 2019.

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to

sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284, 2006.

[65] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL `https://github.com/MadryLab/robustness`.

[66] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[67] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1186–1195, 2018.

[68] Uriel Feige, Yishay Mansour, and Robert Schapire. Learning and inference in the presence of corrupted inputs. In *Conference on Learning Theory*, pages 637–657, 2015.

[69] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*, 2020.

[70] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[71] Amos Fiat and Dmitry Pechyony. Decision trees: More theoretical justification for practical algorithms. In *International Conference on Algorithmic Learning Theory*, pages 156–170, 2004.

[72] Chris Finlay and Adam M Oberman. Scaleable input gradient regularization for adversarial robustness. *arXiv preprint arXiv:1905.11468*, 2019.

[73] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv:1901.10513*, 2019.

[74] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[75] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Momputer and System Sciences*, 55(1):119–139, 1997.

[76] Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. Exkmc: Expanding explainable k-means clustering. *arXiv preprint arXiv:2006.02399*, 2020.

[77] Shivam Garg, Vatsal Sharan, Brian Zhang, and Gregory Valiant. A spectral view of adversarially robust features. In *Advances in Neural Information Processing Systems*, pages 10138–10148, 2018.

[78] Damien Garreau and Ulrike von Luxburg. Explaining the explainer: A first theoretical analysis of lime. *arXiv preprint arXiv:2001.03447*, 2020.

[79] Damien Garreau and Ulrike von Luxburg. Looking deeper into lime. *arXiv preprint arXiv:2008.11092*, 2020.

[80] Geoffrey Gates. The Reduced Nearest Neighbor Rule. *IEEE transactions on information theory*, 18(3):431–433, 1972.

[81] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[82] Antonio Ginart, Melody Y Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012*, 2019.

[83] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Concerence on Learning Representations*, 2015.

[84] Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759, 2014.

[85] Lee-Ad Gottlieb, Aryeh Kontorovich, and Pinhas Nisnevitch. Near-Optimal Sample Compression for Nearest Neighbors. In *Advances in neural information processing systems*, pages 370–378, 2014.

[86] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for PGD-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019.

[87] Minghao Guo, Yuzhe Yang, Rui Xu, and Ziwei Liu. When nas meets robustness: In search of robust architectures against adversarial attacks. *arXiv preprint arXiv:1911.10695*, 2019.

[88] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018. URL `http://www.gurobi.com`.

[89] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

[90] Peter Hart. The Condensed Nearest Neighbor Rule. *IEEE transactions on information theory*, 14(3):515–516, 1968.

[91] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for

image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[93] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2263–2273, 2017.

[94] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017.

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

[96] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[97] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, pages 7267–7275, 2019.

[98] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[99] Kuan-Hao Huang, Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. Improving zero-shot cross-lingual transfer learning via robust training. *arXiv preprint arXiv:2104.08645*, 2021.

[100] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29, 2018.

[101] Angel Hsing-Chi Hwang, Cheng Yao Wang, Yao-Yuan Yang, and Andrea Stevenson Won. Hide and seek: Choices of virtual backgrounds in video chats and their effects on perception. In *ACM on Human-Computer Interaction*, volume 5, pages 1–30, 2021.

[102] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On relating explanations and adversarial examples. In *Advances in Neural Information Processing Systems*, pages 15883–15893, 2019.

[103] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016, 2021.

[104] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus.

arXiv preprint arXiv:1702.08734, 2017.

[105] Matt Jordan, Justin Lewis, and Alexandros G. Dimakis. Provable Certificates for Adversarial Examples: Fitting a Ball in the Union of Polytopes. *arXiv preprint arXiv:1903.08778*, 2019.

[106] Varun Kanade and Adam Kalai. Potential-based agnostic boosting. *Advances in Neural Information Processing Systems*, 22:880–888, 2009.

[107] Daniel Kang, Yi Sun, Dan Hendrycks, Tom Brown, and Jacob Steinhardt. Testing robustness against unforeseen adversaries. *arXiv preprint arXiv:1908.08016*, 2019.

[108] Alex Kantchelian, J Doug Tygar, and Anthony Joseph. Evasion and hardening of tree ensemble classifiers. In *International Conference on Machine Learning*, pages 2387–2396, 2016.

[109] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802*, 2017.

[110] Michael Kearns. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*, 1988.

[111] Michael Kearns and Yishay Mansour. On the boosting ability of top–down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

[112] Fereshte Khani and Percy Liang. Removing spurious features can hurt accuracy and affect groups disproportionately. In *Conference on Fairness, Accountability, and Transparency*, pages 196–205, 2021.

[113] Marc Khoury and Dylan Hadfield-Menell. Adversarial Training with Voronoi Constraints. *Safe Machine Learning workshop at ICLR*, 2019.

[114] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[115] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[116] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. *ICML 2015 Deep Learning Workshop*, 2015.

[117] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

[118] Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

[119] Aryeh Kontorovich and Roi Weiss. A Bayes Consistent 1-NN classifier. In *International Conference on Artificial Intelligence and Statistics*, 2015.

[120] Aryeh Kontorovich, Sivan Sabato, and Roi Weiss. Nearest-neighbor Sample Compression: Efficiency, Consistency, Infinite Dimensions. In *Advances in neural information processing systems*, pages 1573–1583, 2017.

[121] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[122] Bogdan Kulynych, Yao-Yuan Yang, Yaodong Yu, Jarosław Błasiok, and Preetum Nakkiran. What you see is what you get: Distributional generalization for algorithm design in deep learning. *arXiv preprint arXiv:2204.03230*, 2022.

[123] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial examples in the physical world, 2016.

[124] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

[125] Hyun Kwon, Yongchul Kim, Ki-Woong Park, Hyunsoo Yoon, and Daeseon Choi. Multi-targeted adversarial example in evasion attack on deep neural network. *IEEE Access*, 6: 46084–46096, 2018.

[126] Eduardo Laber and Lucas Murtinho. On the price of explainability for some clustering problems. *arXiv preprint arXiv:2101.01576*, 2021.

[127] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[128] Himabindu Lakkaraju and Osbert Bastani. "how do i fool you?" manipulating user trust via misleading black box explanations. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

[129] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. Robust and stable black box explanations. In *International Conference on Machine Learning*, pages 5628–5638, 2020.

[130] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[131] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In {*USENIX*} *Security Symposium*, pages 1605–1622, 2020.

[132] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive gaussian noise. *arXiv preprint arXiv:1809.03113*, 2018.

[133] Xiao-Hui Li, Yuhan Shi, Haoyang Li, Wei Bai, Yuanwei Song, Caleb Chen Cao, and Lei Chen. Quantitative evaluations on saliency methods: An experimental study. *arXiv preprint arXiv:2012.15616*, 2020.

[134] Yiming Li, Baoyuan Wu, Yan Feng, Yanbo Fan, Yong Jiang, Zhifeng Li, and Shutao Xia. Toward adversarial robustness via semi-supervised robust training. *arXiv preprint arXiv:2003.06974*, 2020.

[135] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

[136] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[137] Jinlong Liu, Guoqing Jiang, Yunzhi Bai, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through gsnr of parameters. *arXiv preprint arXiv:2001.07384*, 2020.

[138] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *International Conference on Learning Representations*, 2017.

[139] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[140] Daniel Lowd and Christopher Meek. Adversarial learning. In *SIGKDD*, pages 641–647, 2005.

[141] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[142] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[143] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with Lipschitz functions. *Journal of Machine Learning Research*, 5:669–695, 2004.

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

[145] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Informa-

tion Retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[146] Dina Mardaoui and Damien Garreau. An analysis of lime for text data. *arXiv preprint arXiv:2010.12487*, 2020.

[147] Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. *arXiv preprint arXiv:1909.12180*, 2019.

[148] Yifei Min, Lin Chen, and Amin Karbasi. The curious case of adversarially robust models: More data can help, double descend, or hurt generalization. *arXiv preprint arXiv:2002.11080*, 2020.

[149] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[150] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[151] Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. Explainable k-means and k-medians clustering. In *International Conference on Machine Learning*, pages 7055–7065, 2020.

[152] Michal Moshkovitz, Yao-Yuan Yang, and Kamalika Chaudhuri. Connecting interpretability and robustness in decision trees through separation. *arXiv preprint arXiv:2102.07048*, 2021.

[153] Ketan Mulmuley. On levels in arrangements and voronoi diagrams. *Discrete & Computational Geometry*, 6(3):307–338, 1991.

[154] Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. Convergence of gradient descent on separable data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3420–3428, 2019.

[155] Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

[156] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[157] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. *arXiv preprint arXiv:2007.02923*, 2020.

[158] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop*

on Deep Learning and Unsupervised Feature Learning, 2011.

[159] Robert M Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*, 115(1):39, 1986.

[160] Alexander G Ororbia II, Daniel Kifer, and C Lee Giles. Unifying adversarial training algorithms with data gradient regularization. *Neural computation*, 29(4):867–887, 2017.

[161] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[162] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

[163] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

[164] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[165] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy*, 2016.

[166] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, Abhibhav Garg, and Yen-Chen Lin. Cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2017.

[167] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *ACM ASIA Conference on Computer and Communications Security*, 2017.

[168] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[169] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,

M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[170] Rafael Pinot, Laurent Meunier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif. Theoretical evidence for adversarial robustness through randomization: the case of the exponential family. *arXiv preprint arXiv:1902.01148*, 2019.

[171] Muni Sreenivas Pydi and Varun Jog. Adversarial risk via optimal transport and optimal couplings. *arXiv preprint arXiv:1912.02794*, 2019.

[172] Haifeng Qian and Mark N. Wegman. L2-nonexpansive neural networks. *arXiv preprint arXiv:1802.07896*, 2018.

[173] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pages 13824–13833, 2019.

[174] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International Conference on Machine Learning*, pages 5231–5240, 2019.

[175] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[176] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[177] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

[178] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. *arXiv preprint arXiv:2002.10716*, 2020.

[179] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[180] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[181] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[182] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. *arXiv preprint arXiv:2002.11569*, 2020.

[183] Eitan Richardson and Yair Weiss. A bayes-optimal view on adversarial examples. *arXiv preprint arXiv:2002.08859*, 2020.

[184] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *arXiv preprint arXiv:1711.09404*, 2017.

[185] Jeffrey N Rouder and Roger Ratcliff. Comparing categorization models. *Journal of Experimental Psychology: General*, 133(1):63, 2004.

[186] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[187] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335, 2020.

[188] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[189] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pages 8346–8356, 2020.

[190] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 11289–11300, 2019.

[191] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *arXiv preprint arXiv:2007.08489*, 2020.

[192] Adam N Sanborn, Katherine Heller, Joseph L Austerweil, and Nick Chater. Refresh: A new approach to modeling dimensional biases in perceptual similarity and categorization. *Psychological Review*, 2021.

[193] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859*, 2020.

[194] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *Advances in Neural Information Processing Systems*, pages 4053–4061, 2016.

[195] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*,

2013.

[196] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems 31*, pages 5019–5031, 2018.

[197] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

[198] Terrence J Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

[199] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[200] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. *arXiv preprint arXiv:1905.08232*, 2019.

[201] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018.

[202] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[203] Shai Shalev-Shwartz and Yoram Singer. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. In *21st Annual Conference on Learning Theory, COLT 2008*, 2008.

[204] Ohad Shamir. Gradient methods never overfit on separable data. *arXiv preprint arXiv:2007.00028*, 2020.

[205] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[206] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pages 10802–10813, 2018.

[207] Sahil Singla and Soheil Feizi. Causal imagenet: How to discover spurious features in deep learning? *arXiv preprint arXiv:2110.04301*, 2021.

[208] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness

with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[209] Chawin Sitawarin and David Wagner. On the Robustness of Deep K-Nearest Neighbors. *arXiv preprint arXiv:1903.08333*, 2019.

[210] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[211] Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Improving the Generalization of Adversarial Training with Domain Adaptation. In *International Conference on Learning Representations*, 2019.

[212] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19:2822–2878, 2018.

[213] Megha Srivastava, Tatsunori Hashimoto, and Percy Liang. Robustness to spurious correlations via human annotations. In *International Conference on Machine Learning*, pages 9109–9119, 2020.

[214] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[215] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. *CoRR, abs/1910.06259*, 2019.

[216] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[217] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *European Conference on Computer Vision*, pages 631–648, 2018.

[218] Lu Sun, Mingtian Tan, and Zhe Zhou. A survey of practical adversarial example attacks. *Cybersecurity*, 1(1):1–9, 2018.

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

[220] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and

Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.

[221] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Concerence on Learning Representations*, 2019.

[222] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[223] Berk Ustun and Cynthia Rudin. Optimized risk scores. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1125–1134, 2017.

[224] Berk Ustun and Cynthia Rudin. Learning optimized risk scores. *Journal of Machine Learning Research*, 20(150):1–75, 2019.

[225] Francisco Utrera, Evan Kravitz, N Benjamin Erichson, Rajiv Khanna, and Michael W Mahoney. Adversarially-trained deep nets transfer better. *arXiv preprint arXiv:2007.05869*, 2020.

[226] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020.

[227] Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

[228] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the Robustness of Nearest Neighbors to Adversarial Examples. In *International Conference on Machine Learning*, pages 5120–5129, 2018.

[229] 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. *arXiv preprint arXiv:1801.10578*, 2018.

[230] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[231] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160: 53–62, 2015.

[232] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

[233] Kai Y Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training

for faster adversarial robustness verification via inducing relu stability. In *International Concerence on Learning Representations*, 2019.

[234] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009.

[235] Yao-Yuan Yang and Kamalika Chaudhuri. Understanding rare spurious correlations in neural networks. *arXiv preprint arXiv:2202.05189*, 2022.

[236] Yao-Yuan Yang, Kuan-Hao Huang, Chih-Wei Chang, and Hsuan-Tien Lin. Cost-sensitive reference pair encoding for multi-label learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 143–155, 2018.

[237] Yao-Yuan Yang, Yi-An Lin, Hong-Min Chu, and Hsuan-Tien Lin. Deep learning with a rethinking structure for multi-label classification. In *Asian Conference on Machine Learning*, pages 125–140, 2019.

[238] Yao-Yuan Yang, Cyrus Rashtchian, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Robustness and generalization to nearest categories. *arXiv preprint arXiv:2011.08485*, 2020.

[239] Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri. Robustness for non-parametric classification: A generic attack and defense. In *International Conference on Artificial Intelligence and Statistics*, pages 941–951, 2020.

[240] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *Advances in Neural Information Processing Systems*, pages 8588–8601, 2020.

[241] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. Torchaudio: Building blocks for audio and speech processing. In *International Conference on Acoustics, Speech and Signal Processing*, pages 6982–6986, 2022.

[242] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[243] Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*, 2019.

[244] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *Interna-*

tional Conference on Machine Learning, 2019.

[245] Jingfeng Zhang, Bo Han, Gang Niu, Tongliang Liu, and Masashi Sugiyama. Where is the bottleneck of adversarial learning with unlabeled data? *arXiv preprint arXiv:1911.08696*, 2019.

[246] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. *arXiv preprint arXiv:2002.11242*, 2020.

[247] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology*, 11(3):1–41, 2020.

[248] Xiao Zhang, Jinghui Chen, Quanquan Gu, and David Evans. Understanding the intrinsic robustness of image distributions using conditional generative models. *arXiv preprint arXiv:2003.00378*, 2020.

[249] Chunting Zhou, Xuezhe Ma, Paul Michel, and Graham Neubig. Examining and combating spurious features under distribution shift. In *International Conference on Machine Learning*, volume 139, pages 12857–12867, 2021.

[250] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.