UNIVERSITY OF CALIFORNIA

Los Angeles

Evaluating and Understanding Adversarial Robustness in Deep Learning

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

by

Jinghui Chen

2021

ABSTRACT OF THE DISSERTATION

Evaluating and Understanding Adversarial
Robustness in Deep Learning

by

Jinghui Chen
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Quanquan Gu, Chair

Deep Neural Networks (DNNs) have made many breakthroughs in different areas of artificial intelligence. However, recent studies show that DNNs are vulnerable to adversarial examples. A tiny perturbation on an image that is almost invisible to human eyes could mislead a well-trained image classifier towards misclassification. This raises serious security concerns and trustworthy issues towards the robustness of Deep Neural Networks in solving real world challenges. Researchers have been working on this problem for a while and it has further led to a vigorous arms race between heuristic defenses that propose ways to defend against existing attacks and newly-devised attacks that are able to penetrate such defenses. While the arm race continues, it becomes more and more crucial to accurately evaluate model robustness effectively and efficiently under different threat models and identify those "falsely" robust models that may give us a false sense of robustness. On the other hand, despite the fast development of various kinds of heuristic defenses, their practical robustness is still far from satisfactory, and there are actually little algorithmic improvements in terms of defenses during recent years. This suggests that there still lacks further understandings toward the fundamentals of adversarial robustness in deep learning, which might prevent us from designing more powerful defenses.

The overarching goal of this research is to enable accurate evaluations of model robustness under different practical settings as well as to establish a deeper understanding towards other factors in the machine learning training pipeline that might affect model robustness. Specifically, we develop efficient and effective Frank-Wolfe attack algorithms under white-box and black-box settings and a hard-label adversarial attack, RayS, which is capable of detecting "falsely" robust models. In terms of understanding adversarial robustness, we propose to theoretically study the relationship between model robustness and data distributions, the relationship between model robustness and model architectures, as well as the relationship between model robustness and loss smoothness. The techniques proposed in this dissertation form a line of researches that deepens our understandings towards adversarial robustness and could further guide us in designing better and faster robust training methods.

The dissertation of Jinghui Chen is approved.

University of California, Los Angeles

2021

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am fortunate to have the support, guidance, and collaborations from a large group of people, from my advisor, colleagues to friends and family, during my entire six-year Ph.D. journey. I greatly appreciate all their help and support.

I want to sincerely thank my advisor, Prof. Quanquan Gu, who is modest and industrious while being super smart. Prof. Quanquan Gu has led me through this journey and turn me from a fresh undergraduate student to an experienced researcher. I am grateful for all the training, mentoring and teaching from him, especially during the early years of my Ph.D. journey. Without them, I might still be the young man who knows nothing about machine learning research. I am also very grateful that Prof. Quanquan Gu offered me the opportunity to transfer with him and came to UCLA to continue my Ph.D. journey and my academic career.

I also want to thank all my mentors and supervisors during my several industrial internships. I gained tons of valuable experiences and knew many awesome people during my first internship at IBM T.J. Watson Research Center. My mentors, Dr. Charu Aggarwal and Dr. Saket Sathe, are nice, knowledgeable, excellent in outlier detection related research. Their guidance helped me a lot and I really enjoyed the time at IBM. For my second internship at JD AI Research, I want to thank Dr. Jinfeng Yi for providing me the remote internship opportunity and get me on board with the adversarial machine learning research, which is the major topic for this dissertation. As the only remote intern that year, I also want to thank Dr. Jing Huang for hosting me in the JD Silicon Valley Research Center. My internship at Twitter is more engineering-oriented, and I am fortunate to have Dr. Yao Wu as my mentor who helped me a lot during the internship and gave me so many useful tips in working and living. My last internship was at Microsoft Research, though being fully remote due to COVID-19. I am fortunate to get this opportunity and have Dr. Yu Cheng and Dr. Zhe Gan as my mentors.

I am also grateful to all other wonderful collaborators during my Ph.D. research. Prof. David Evans is meticulous about the research he is doing. His suggestions on our paper and my research statements are very helpful and constructive. Prof. Jian Ma is easy-going, humor and collaboration with him is always an enjoyable procedure. I would also like to thank my other dissertation committee members, Prof. Adnan Darwiche, Prof. Cho-Jui Hsieh, Prof, Deanna Needell, for all their helpful feedback and suggestions on my Ph.D. research and dissertation work.

I am also fortunate to be part of our Statistical Machine Learning Lab and have so many talented lab mates, Pan Xu, Lu Tian, Lingxiao Wang, Dongruo Zhou, Difan Zou, Yuan Cao, Yue Wu, Jiafan He, Weitong Zhang, Zixiang Chen, who have built a great environment for research. I have learned and benefited a lot from discussions with them. Special thanks to Xiao Zhang and Yaodong Yu, who stayed with me back in UVa and spent together that wonderful year.

Also many thanks to my friends at UCLA and UVa for the company. Thanks to Huazheng Wang, Qingyun Wu, Zhanyu Yang, Tianlu Wang, Dezhi Hong, Zeya Chen, Jianfeng Chi, Zhiqiu Jiang, Yinghan Wang, Renqing Cai, Fnu Suya, Weilin Xu, Bo Yang, Yue Hu, Xin Wen, Frank Feng, Minbiao Han, Pingyu He, Jiekun Yang, Jinghe Zhang, Mengmeng Ye, Feiyu Jin, Jian Kang, Zuo Li, Haoyi Liang, Haoran Liu, Jisi Liu, Fuwen Tan, Xiang Wan, Jie Wang, Xintian Wang, Yiran Wang, Jizhao Zang in UVa, and Minhao Cheng, Xuanqing Liu, Yuanhao Xiong, Xiangning Chen, Xu Xie, Zewen Yuan, Boyang Fu in UCLA. Thank you all for your friendship and support. Also thanks to my old friends from high school as well as new friends met during internships: Houping Xiao, Chunxi Yu, Yumeng Tao, Xinyu Hu, Tian Jin, Zhijun Yin, Qinglong Li, Lifu Huang, Tianjie Wang, Yifan Tian, Yamin Li, Nan Hu, Rui Xu, Wei Xia, Zhekang Peng and many others.

Thanks to those who helped a lot during my job search. Special thanks to Prof. Fenglong Ma, and Prof. Xiaowei Jia and Prof. Jiebo Luo, who provided me with so many useful tips and advice and helped me a lot, especially during this hard year.

Finally, I owe my deepest gratitude to my family, my parents, my grandma, and Lu. I would like to thank Lu for the love, support, company and all the moments together during the past years and I look forward to the many years to come. And I want to thank my parents and my grandparents for the unconditioned love and support, which is my source of power through this journey.

VITA

2015	B.S. (Electronic Engineering and Information Science), University of Science and Technology of China (USTC).
2016	Research Intern, IBM T.J Watson Research Center.
2015–2017	Teaching Assistant, Systems and Information Engineering Department, University of Virginia.
2017–2018	Research Assistant, Computer Science Department, University of Virginia.
2018	Research Intern, JD.COM Silicon Valley Research Center.
2018–2019	Teaching Assistant, Computer Science Department, University of Virginia.
2019	Machine Learning Intern, Twitter.
2019–2020	Research Assistant, Computer Science Department, UCLA.
2020	Ph.D. Candidate in Computer Science, UCLA.
2020	Research Intern, Microsoft.
2020–present	Teaching Assistant, Computer Science Department, UCLA.

PUBLICATIONS

Dongruo Zhou*, **Jinghui Chen***, Yuan Cao*, Yiqi Tang, Ziyan Yang, Quanquan Gu, On the Convergence of Adaptive Gradient Methods for Nonconvex Optimization, NeurIPS 2020 Workshop on Optimization for Machine Learning.

Jinghui Chen, Quanquan Gu, RayS: A Ray Searching Method for Hard-label Adversarial Attack, in Proc of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), San Diego, CA, USA 2020.

Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao and Quanquan Gu, Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks, in Proc. of 29th International Joint Conference on Artificial Intelligence (IJCAI), Yokohama, Japan , 2020.

Xiao Zhang*, **Jinghui Chen***, Quanquan Gu and David Evans, Understanding the Intrinsic Robustness of Image Distributions using Conditional Generative Models, In Proc of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS), Palermo, Sicily, Italy, 2020.

Jinghui Chen, Dongruo Zhou, Jinfeng Yi, Quanquan Gu, A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks, in Proc. of the 34th Conference on Artificial Intelligence (AAAI), New York, New York, USA, 2020.

Pan Xu*, **Jinghui Chen***, Quanquan Gu, Global Convergence of Langevin Dynamics Based Algorithms for Nonconvex Optimization, In Proc. of the 32nd Advances in Neural Information Processing Systems (NeurIPS), Montréal, Canada, 2018.

Jinghui Chen, Pan Xu, Lingxiao Wang, Jian Ma, Quanquan Gu, Covariate Adjusted Precision Matrix Estimation via Nonconvex Optimization, in Proc. of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018.

Jinghui Chen, Quanquan Gu, Fast Newton Hard Thresholding Pursuit for Sparsity Constrained Nonconvex Optimization, in Proc of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Halifax, Nova Scotia, Canada, 2017.

Jinghui Chen, Saket Sathe, Charu Aggarwal, Deepak Turaga, Outlier Detection with Autoencoder Ensembles, in Proc of 2017 SIAM International Conference on Data Mining

(SDM), Houston, Texas, USA, 2017.

Jinghui Chen, Quanquan Gu, Stochastic Block Coordinate Gradient Descent for Sparsity Constrained Optimization, in Proc of the 32nd International Conference on Uncertainty in Artificial Intelligence (UAI), New York / New Jersey, USA, 2016.

Florian Baumann, **Jinghui Chen**, Karsten Vogt, Bodo Rosenhahn, Improved threshold Selection by using Calibrated Probabilities for Random Forest Classifiers, 12th Conference on Computer and Robot Vision (CRV), Halifax, Nova Scotia, Canada, 2015.

CHAPTER 1

Introduction

1.1 Overview

Deep Neural Networks (DNNs) have made many breakthroughs in different areas of artificial intelligence such as image classification [KSH12, HZRS16a], object detection [RHGS15, Gir15], and speech recognition [MDH+12, BCS+16]. However, recent studies show that deep neural networks can be vulnerable to adversarial examples [SZS+13, GSS15] – a tiny perturbation on an image that is almost invisible to human eyes could mislead a well-trained image classifier towards misclassification. Figure 1.1 demonstrates an adversarial example generated against GoogLeNet [SLJ+15]. Soon later this is proved to be not a coincidence: similar phenomena have been observed in other problems such as speech recognition [CMV+16], visual QA [XCL+17], image captioning [CZC+17], machine translation [CYZ+18], reinforcement learning [PTL+18], and even on systems that operate in the physical world [KGB16].

Depending on how much information an adversary can access to, attacks can be classified into three classes: white-box attack [SZS+13, GSS15], black-box attack [PMG16, CZS+17], and hard-label attack [CLC+19, CSC+20, CJW19]. In the white-box setting, the adversary has full access to the target model, including the model architecture, the model weights and all the network outputs including both logits output and prediction labels. Therefore, it is possible to perform back-propagation on the neural network in this setting. Specifically, suppose $f_{\boldsymbol{\theta}}$ denotes the neural network parameterized by $\boldsymbol{\theta}$, ℓ denotes the loss function, for some specific data example $(\mathbf{x}, y)$, we can generate its adversarial example by solving the

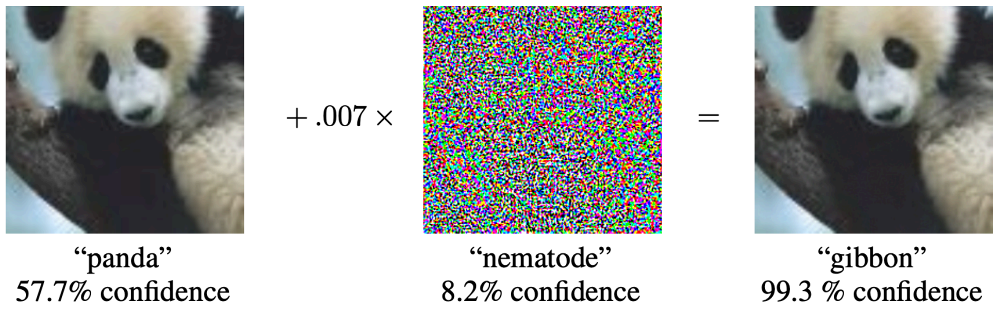


Figure 1.1: A demonstration of adversarial example generated against GoogLeNet [SLJ+15] on ImageNet. By adding an imperceptibly small noise we manage to change the classification result significantly [GSS15].

following maximization problem

$$\max_{\boldsymbol{\delta} \in \boldsymbol{\Delta}} \ \ell(\mathbf{x} + \boldsymbol{\delta}, y; \boldsymbol{\theta}),$$

where the adversarial perturbation $\boldsymbol{\delta}$ needs to satisfy some certain constraint such that the perturbation is small and invisible to human eyes, e.g., L_{inf} norm ϵ-ball constraint. The black-box setting, on the other hand, treats the major part of the neural network as a black box, i.e., the adversary can only access the logit output and prediction labels of the target model of a given query but not its internal configurations. This prevents people from using back-propagation under this setting as the neural network weights are no longer available here. The most practical setting for adversarial attacks would be the hard-label setting, where the adversary is only allowed to access the final prediction label of a given query but nothing else. It treats the entire neural network as a black box. Figure 1.2 provides an illustration of the above mentioned three different settings for adversarial attacks. It is easy to observe that from white-box attack to black-box attack, to hard-label attack, the attack problem becomes harder and harder to solve due to more and more limited information access.

On the other hand, people have been trying to defend against adversarial examples since this phenomenon has been discovered. During the process, many heuristic defenses have been

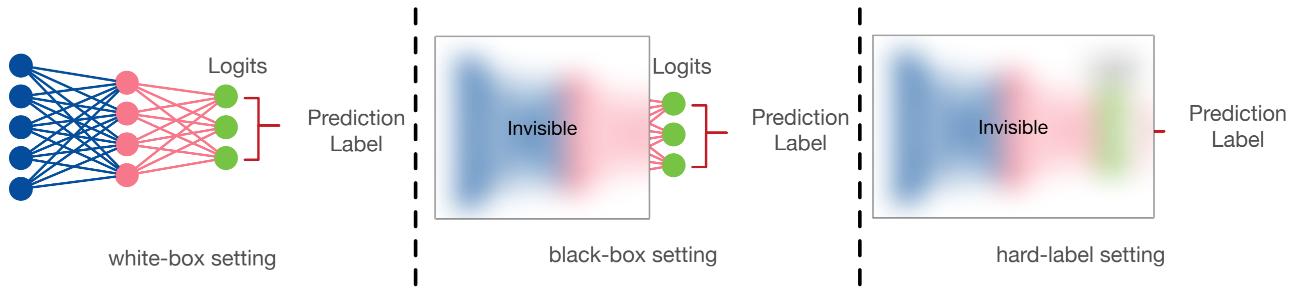


Figure 1.2: An illustration of three different settings for adversarial attacks.

proposed and many of them later were shown to be not effective or can be broken by newly-devised attacks. Among those defenses, one typical and effective way is through adversarial training, meaning one can train on adversarial examples for defenses. Specifically, standard adversarial training proposes to solve the following min-max optimization problem.

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \max_{\boldsymbol{\delta} \in \boldsymbol{\Delta}} \ \ell(\mathbf{x}_i + \boldsymbol{\delta}, y_i; \boldsymbol{\theta}).$$

While adversarial training indeed largely improves model robustness compared to standard training, its practical robustness is still far from satisfactory at the current stage, and there are actually little algorithmic improvements in terms of defenses during the recent years since adversarial training is invented [RWK20a].

1.2 Challenges

Current adversarial machine learning research still faces numerous challenges. Here we pick several typical challenges in evaluating and understanding adversarial robustness and discuss them in detail.

1.2.1 Evaluating Adversarial Robustness

Machine learning models can be attacked in various settings, depending on how much information is available to the adversary. The commonly imposed yet the simplest setting is the white-box attack, where the attacker can access every detail about the target model. However, such a level of information availability to the target model is usually unrealistic in practice. In most scenarios, the attacker can only query the target model for the logits outputs (black-box attack) or even prediction labels alone (hard-label attack), which makes the attack problem more practical but challenging. Most attack algorithms targeting such settings suffer from high query complexities, i.e., a large number of queries are needed for one successful attack, thereby limiting their practical applicability.

Therefore, the natural question to ask here is:

Can we develop more query-efficient adversarial attacks for evaluating model robustness under practical settings?

Another important problem in evaluating model robustness is that the attack effectiveness. Recent studies [CW17, ACW18] have shown that many defenses can hide or distort the gradient to fail those gradient-based attacks such as PGD but do not actual improve model robustness. Those "obfuscated-gradient" defenses can still be vulnerable to specialized attacks. In other words, PGD attack can be non-effective for those "obfuscated-gradient" defenses and therefore, it is important to ask:

Can we develop methods that can accurately evaluate model robustness and identify those possible "falsely robust" models?

Two of my works (introduce in Chapters 2 and 3) aim at solving the above important questions.

1.2.2 Understanding Adversarial Robustness

So far, the emerging new attacks always take an early lead to defense strategies [ACW18], and the root causes of adversarial vulnerability still remain unclear. To mitigate the situation and develop truly robust machine learning models, I am actively working on understanding the rationale behind model robustness and making fundamental improvements upon it.

There are actually many questions that remain unanswered in the field of adversarial robustness. Here we typically focused on the following topics:

Have we reach the fundamental limit of (the maximum achievable) adversarial robustness on real data distributions?

This is a natural question to ask since we have reached a bottleneck in improving model robustness. Witnessing such difficulties in constructing robust classifiers, a line of recent works [GMF$^+$18, FFF18a, MDM19, SHS$^+$19] show that no adversarially robust classifiers exist for an assumed metric probability space, as long as the perturbation strength is sublinear in the typical norm of the inputs, by imposing different assumptions on the underlying data distributions. Although such impossibility results seem disheartening to the goal of building robust classifiers, it remains unknown to what extent real image distributions satisfy the assumptions needed to obtain these results. In particular, we are interested in that, on well-behaved data distributions, have we reached the fundamental limit of adversarial robustness?

How does network architecture (network width) affect adversarial robustness?

One common belief in the community is that wider models usually enjoy better adversarial robustness as it has larger model capacity. However, this belief has never come under scrutiny, and we are still not crystal clear on whether network width can really help adversarial robustness or not, as well as the underlying reasons behind it. Is it possible that the increased network width only helps natural generalization but may hurt robustness? This remains a question that needs to be carefully addressed.

Will smoother loss actually help efficient adversarial training?

Adversarial training methods, despite being effective in improving model robustness, are notoriously inefficient to train. Recently, efficient robust training methods such as Fast Adversarial Training [WRK20] is proposed, however, its performance still lacks stability and has a gap between the current state-of-the-art methods. Fast Adversarial Training [WRK20] works as performing randomized smoothing on the loss objectives and therefore, this intrigues us to study whether smoother loss actually helps efficient robust training.

1.3 Contributions

Two of my works aim at improving query efficiency by eliminating unnecessary computations in the attack algorithms: I proposed a new projection-free black-box attack, motivated by the observation that the projection step in traditional attack algorithms leads to inefficient updates. I developed a novel search-based hard-label attack algorithm, which discretizes the continuous perturbation space and eliminates all unnecessary search directions via a specially designed check step. Furthermore, the proposed attack makes a significant advance over traditional attack algorithms by defining a novel robustness metric to detect *falsely* robust models. Such models could resist traditional white-box or black-box attacks but fail on specially adapted attacks [ACW18]. My study on evaluating model robustness also contributes to the community: I built a leaderboard on Github [1] based on the new metric to evaluate the robustness of the state-of-the-art models.

I also studied the crucial factors influencing model robustness such as data distribution, model architecture and loss smoothness: I investigated the property of real image distributions, based on which I further derived the robustness upper bound that a general class of models (e.g. neural networks) can achieve. My findings demonstrate a large gap between the theoretical upper bound and the empirical model robustness on real image distributions, which implies a large room for further robustness improvements of current models. I studied the relationship between neural network width and adversarial robustness, based on a novel metric called perturbation stability. The empirical results showed that larger network width leads to better natural accuracy but worse perturbation stability, which contradicts the common belief of wider networks always helping model robustness. I further theoreti-

[1]`https://github.com/uclaml/RayS`

cally justified the claim by leveraging recent results on neural tangent kernels [JHG18]. I also studied the role of loss smoothness in robust training methods. In particular, I proposed a new perspective that views the random initialization in Fast Adversarial Training [WRK20] as performing randomized smoothing on the loss objectives. Based on this perspective, I proposed a novel smoothing strategy, called backward smoothing, which significantly improves both model robustness and stability over single-step robust training methods. These works uncover intrinsic properties of adversarial robustness and provide insights on understanding and enhancing model robustness.

1.4 Organization of the Dissertation

The rest of the dissertation is organized and presented as follows.

1. In Chapter 2, we introduce our attempt for effective and efficient adversarial attacks under white-box and black-box settings. The developed new Frank-Wolfe based adversarial attack framework benefits from its projection-free nature and the use of the momentum mechanism. And it also comes with theoretical convergence guarantees.

2. In Chapter 3, we introduce a hard-label attack, RayS, which only relies on the hard-label output of the target model. RayS attack is much more effective and efficient compared with previous hard-label attacks on L_∞ norm threat models. Moreover, it could also be used as a sanity check for possible "falsely" robust models.

3. In Chapter 4, we aim to understand the fundamental limit of robust training on real image distributions[2]. We proved a fundamental bound on intrinsic robustness and observed a large gap between the theoretical intrinsic robust limit and the best robustness achieved by state-of-the-art robust classifiers.

[2]This is joint work with Xiao Zhang, who equally contributes to the work described in this Chapter.

4. In Chapter 5, we study whether wider networks always lead to better model robustness by understanding the relationship between network width and adversarial robustness[3]. We present both empirical and theoretical studies to show that larger network width may not always help model robustness.

5. In Chapter 6, we develop a new understanding towards Fast Adversarial Training, by viewing random initialization as performing randomized smoothing. We also propose a new initialization strategy, backward smoothing, to significantly improves both stability and model robustness over single-step robust training methods.

6. Finally, we conclude the dissertation by summarizing the contributions and elaborate the potential research directions in Chapter 7.

[3]This is joint work with Boxi Wu, who equally contributes to the work described in this Chapter.

CHAPTER 2

A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks

2.1 Introduction

Deep Neural Networks (DNNs) have made many breakthroughs in different areas of artificial intelligence such as image classification [KSH12, HZRS16a], object detection [RHGS15, Gir15], and speech recognition [MDH$^+$12, BCS$^+$16]. However, recent studies show that deep neural networks are vulnerable to adversarial examples [SZS$^+$13, GSS15] – a tiny perturbation on an image that is almost invisible to human eyes could mislead a well-trained image classifier towards misclassification. Soon later this is proved to be not a coincidence in image classification: similar phenomena have been observed in other problems such as speech recognition [CMV$^+$16], visual QA [XCL$^+$17], image captioning [CZC$^+$17], machine translation [CYZ$^+$18], reinforcement learning [PTL$^+$18], and even on systems that operate in the physical world [KGB16].

Depending on how much information an adversary can access, adversarial attacks can be classified into two classes: white-box attack [SZS$^+$13, GSS15] and black-box attack [PMG16, CZS$^+$17]. In the white-box setting, the adversary has full access to the target model, while in the black-box setting, the adversary can only access the input and output of the target model but not its internal configurations.

Several optimization-based methods have been proposed for the white-box attack. One of the first successful attempts is the FGSM method [GSS15], which works by linearizing the network loss function. CW method [CW17] further improves the attack effectiveness by designing a regularized loss function based on the logit-layer output of the network and optimizing the loss by Adam [KB15]. Even though CW largely improves the effectiveness, it requires a large number of gradient iterations to optimize the distortion of the adversarial examples. Iterative gradient (steepest) descent based methods such as PGD [MMS+18] and I-FGSM [KGB16] can achieve relatively high attack success rates within a moderate number of iterations. However, they tend to generate adversarial examples near or upon the boundary of the perturbation set, due to the projection nature of the algorithm. This leads to large distortion in the resulting adversarial examples.

In the black-box attack, since one needs to make gradient estimations in such a setting, a large number of queries are required to perform a successful black-box attack, especially when the data dimension is high. A naive way to estimate gradient direction is to perform finite difference approximation on each dimension [CZS+17]. This would take $O(d)$ queries to perform one full gradient estimation where d is the data dimension and therefore result in inefficient attacks. For example, attacking a $299 \times 299 \times 3$ ImageNet [DDS+09] image may take hundreds of thousands of queries. This significantly limits the practical usefulness of such algorithms since they can be easily defeated by limiting the number of queries that an adversary can make to the target model. Although recent studies [IEA+18, IEM19] have improved the query complexity by using Gaussian sensing vectors or gradient priors, due to the inefficiencies of PGD framework, there is still room for improvements.

In this paper, we propose efficient and effective optimization-based adversarial attack algorithms based on a variant of Frank-Wolfe algorithm. We show in theory that the proposed attack algorithms are efficient with a guaranteed convergence rate. The empirical results also verify the efficiency and effectiveness of our proposed algorithms.

In summary, we make the following main contributions:

1. We develop a new Frank-Wolfe based projection-free attack framework with a mo-

mentum mechanism. The framework contains an iterative first-order white-box attack algorithm which admits the fast gradient sign method (FGSM) as a one-step special case, and also a corresponding black-box attack algorithm that adopts zeroth-order optimization with two sensing vector options (either from the Euclidean unit sphere or from the standard Gaussian distribution).

2. We prove that the proposed white-box and black-box attack algorithms with momentum mechanism enjoy an $O(1/\sqrt{T})$ convergence rate in the nonconvex setting. Compared with existing analyses of Frank-Wolfe for nonconvex optimization [LJ16, RSPS16, BG18], we use momentum in our algorithm for both white-box and black-box attacks and therefore our analysis is more involved. To the best of our knowledge, the convergence of Frank-Wolfe with momentum in the nonconvex setting has never been established before, which is of independent interest. We also show that the query complexity of the proposed black-box attack algorithm is linear in data dimension d.

3. Our experiments on MNIST and ImageNet datasets show that (i) the proposed white-box attack algorithm has better distortion and is more efficient than all the state-of-the-art white-box attack baseline algorithms, and (ii) the proposed black-box attack algorithm is highly query efficient and achieves the highest attack success rate among other baselines.

2.2 Related Work

There is a large body of work on adversarial attacks. In this section, we review the most relevant work in both white-box and black-box attack settings, as well as the non-convex Frank-Wolfe optimization.

White-box Attacks: [SZS$^+$13] proposed to use a box-constrained L-BFGS algorithm for conducting white-box attacks. [GSS15] proposed the Fast Gradient Sign Method (FGSM) based on linearization of the network as a simple alternative to L-BFGS. [KGB16] proposed to iteratively perform the one-step FGSM [GSS15] algorithm and clips the adversarial point back to the distortion limit after every iteration. It is called Basic Iterative Method (BIM) or I-FGM in the literature. [MMS$^+$18] showed that for the L_∞ norm case, BIM/I-FGM is almost[1] equivalent to Projected Gradient Descent (PGD), which is a standard tool for constrained optimization. [PMJ$^+$16] proposed JSMA to greedily attack the most significant pixel based on the Jacobian-based saliency map. [MDFF16] proposed attack methods by projecting the data to the closest separating hyperplane. [CW17] introduced the so-called CW attack by proposing multiple new loss functions for generating adversarial examples. [CSZ$^+$17] followed CW's framework and use an Elastic Net term as the distortion penalty. [DLP$^+$18] proposed MI-FGSM to boost the attack performances using momentum.

Black-box Attacks: One popular family of black-box attacks [HT17, PMG16, PMG$^+$17] is based on the transferability of adversarial examples [LCLS18, BHLS17], where an adversarial example generated for one DNN may be reused to attack other neural networks. This allows the adversary to construct a substitute model that mimics the targeted DNN, and then attack the constructed substitute model using white-box attack methods. However, this type of attack algorithm usually suffers from large distortions and relatively low success rates [CZS$^+$17]. To address this issue, [CZS$^+$17] proposed the Zeroth-Order Optimization (ZOO) algorithm that extends the CW attack to the black-box setting and uses a zeroth-order optimization approach to conduct the attack. Although ZOO achieves much higher attack success rates than the substitute model-based black-box attacks, it suffers from a poor query complexity since its naive implementation requires estimating the gradients of all the coordinates (pixels) of the image. To improve its query complexity, several approaches have been proposed. For example, [TTC$^+$18] introduces an adaptive random gradient estimation algorithm and a well-trained Autoencoder to speed up the attack process. [IEA$^+$18]

[1]Standard PGD in the optimization literature uses the exact gradient to perform the update step while PGD [MMS$^+$18] is actually the steepest descent [BV04] with respect to L_∞ norm.

and [LCLS18] improved ZOO's query complexity by using Natural Evolutionary Strategies (NES) [WSG$^+$14, SHC$^+$17] and active learning, respectively. [IEM19] further improve the performance by considering the gradient priors. [LLW$^+$19] proposed to learn the distributions of adversarial examples to achieve better black-box attack performance. [MAS19] re-formulated the black-box attack problem as a discrete surrogate optimization problem and used a combinatorial search algorithm to improve the query efficiency.

Non-convex Frank-Wolfe Algorithms: The Frank-Wolfe algorithm [FW56], also known as the conditional gradient method, is an iterative optimization method for general constrained optimization problem. [Jag13] revisited the Frank-Wolfe algorithm in 2013 and provided a stronger and more general convergence analysis in the convex setting. [YZS17] proved the first convergence rate for the Frank-Wolfe type algorithm in the non-convex setting. [LJ16] provided the convergence guarantee for the Frank-Wolfe algorithm in the non-convex setting with adaptive step sizes. [RSPS16] further studied the convergence rate of non-convex stochastic Frank-Wolfe algorithm in the finite-sum optimization setting. Very recently, [SJ17] proposed to use Frank-Wolfe for distributionally robust training [SND18]. [BG18] proved the convergence rate for zeroth-order nonconvex Frank-Wolfe algorithm using one-side finite difference gradient estimator with standard Gaussian sensing vectors.

2.3 Methodology

2.3.1 Notation

Throughout the paper, scalars are denoted by lower case letters, vectors by lower case bold face letters and sets by calligraphy upper cae letters. For a vector $\mathbf{x} \in \mathbb{R}^d$, we denote the L_p norm of $\mathbf{x}$ by $\|\mathbf{x}\|_p = (\sum_{i=1}^d x_i^p)^{1/p}$. Specially, for $p = \infty$, the L_∞ norm of $\mathbf{x}$ by $\|\mathbf{x}\|_\infty = \max_{i=1}^d |\theta_i|$. We denote $\mathcal{P}_{\mathcal{X}}(\mathbf{x})$ as the projection operation of projecting vector $\mathbf{x}$ into the set $\mathcal{X}$.

2.3.2 Problem Formulation

According to the attack purposes, attacks can be divided into two categories: *untargeted attack* and *targeted attack*.

In particular, the untargeted attack aims to turn the prediction into any incorrect label, while the targeted attack, requires misleading the classifier to a specific target class. In this work, we focus on the strictly harder targeted attack setting [CW17, IEA$^+$18]. It is worth noting that our proposed algorithm can be extended to untargeted attacks straightforwardly. To be more specific, let us define $\ell(\mathbf{x}, y)$ as the classification loss function of the targeted DNN with an input $\mathbf{x} \in \mathbb{R}^d$ and a corresponding label y. For targeted attacks, we aim to minimize $\ell(\mathbf{x}, y_{\text{tar}})$ to learn an adversarial example that will be misclassified to the target class y_{tar}. In the rest of this paper, let $f(\mathbf{x}) = \ell(\mathbf{x}, y_{\text{tar}})$ be the attack loss function for simplicity, and the corresponding targeted attack problem [2] can be formulated as the following optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to} \quad \|\mathbf{x} - \mathbf{x}_{\text{ori}}\|_p \leq \epsilon. \tag{2.3.1}$$

Evidently, the constraint set $\mathcal{X} := \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_{\text{ori}}\|_p \leq \epsilon\}$ is a bounded convex set when $p \geq 1$. Note that even though we mainly focus on the most popular L_∞ attack case in this paper, our proposed methods can easily extend to the general $p \geq 1$ case.

2.3.3 Frank-Wolfe vs. PGD

Although PGD can achieve a relatively high attack success rate within moderate iterates, the multi-step update formula requires an additional projection step at each iteration to keep the iterates within the constraint set. This tends to cause the generated adversarial examples near or upon the boundary of the constraint set, and leads to relatively large distortion. This motivates us to use Frank-Wolfe based optimization algorithm [FW56]. Different from PGD,

[2] Note that there is usually an additional constraint on the input variable $\mathbf{x}$, e.g., $\mathbf{x} \in [0, 1]^n$ for normalized image inputs.

Frank-Wolfe algorithm is projection-free as it calls a Linear Minimization Oracle (LMO) over the constraint set $\mathcal{X}$ at each iteration, i.e.,

$$\text{LMO} \in \operatorname*{argmin}_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{x}, \nabla f(\mathbf{x}_t) \rangle.$$

The LMO can be seen as the minimization of the first-order Taylor expansion of $f(\cdot)$ at point $\mathbf{x}_t$:

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}_t) + \langle \mathbf{x} - \mathbf{x}_t, \nabla f(\mathbf{x}_t) \rangle.$$

By calling LMO, Frank Wolfe solves the linear problem in $\mathcal{X}$ and then performs weighted average with the previous iterate to obtain the final update formula.

Comparing the two methods, PGD is a more “aggressive” approach. It first takes a step towards the negative gradient direction while ignoring the constraint to get a new point (often outside the constraint set), and then correct the new point by projecting it back into the constraint set. In sharp contrast, Frank-Wolfe is more “conservative” as it always keeps the iterates within the constraint set. Therefore, it avoids projection and can lead to better distortion.

2.3.4 Frank-Wolfe White-box Attacks

The proposed Frank-Wolfe based white-box attack algorithm is shown in Algorithm 1, which is built upon the classic Frank-Wolfe algorithm. The key difference between Algorithm 1 and the classic Frank-Wolfe algorithm is in Line 4, where an additional momentum term $\mathbf{m}_t$ is introduced. The momentum term $\mathbf{m}_t$ will help stabilize the LMO direction and leads to empirically accelerated convergence of Algorithm 1.

The LMO solution itself can be expensive to obtain in general. Fortunately, for the constraint set $\mathcal{X}$ defined in (2.3.1), the corresponding LMO has a closed-form solution. Here

Algorithm 1 Frank-Wolfe White-box Attack Algorithm

```
1: input: number of iterations $T$, step sizes $\{\gamma_t\}$;
2: $\mathbf{x}_0 = \mathbf{x}_{\text{ori}}, \mathbf{m}_{-1} = \nabla f(\mathbf{x}_0)$
3: for $t = 0, \ldots, T-1$ do
4:     $\mathbf{m}_t = \beta \cdot \mathbf{m}_{t-1} + (1-\beta) \cdot \nabla f(\mathbf{x}_t)$
5:     $\mathbf{v}_t = \text{argmin}_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{x}, \mathbf{m}_t \rangle$  // LMO
6:     $\mathbf{d}_t = \mathbf{v}_t - \mathbf{x}_t$
7:     $\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t \mathbf{d}_t$
8: end for
9: output:  $\mathbf{x}_T$
```

we provide the closed-form solution of LMO (Line 5 in Algorithm 1) for L_∞ norm case [3]:

$$\mathbf{v}_t = -\epsilon \cdot \text{sign}(\mathbf{m}_t) + \mathbf{x}_{\text{ori}}.$$

Note that if we write down the full update formula at each iteration in Algorithm 1, it becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \epsilon \cdot \text{sign}(\mathbf{m}_t) - \gamma_t (\mathbf{x}_t - \mathbf{x}_{\text{ori}}). \tag{2.3.2}$$

Intuitively speaking, the term $-\gamma_t(\mathbf{x}_t - \mathbf{x}_{\text{ori}})$ enforces $\mathbf{x}_t$ to be close to $\mathbf{x}_{\text{ori}}$ for all $t = 1, \ldots, T$, which encourages the adversarial example to have a small distortion. This is the key advantage of Algorithm 1.

Comparison with FGSM: When $T = 1$, substituting the above LMO solutions into Algorithm 1 yields the final update of $\mathbf{x}_1 = \mathbf{x}_0 - \gamma_t \epsilon \cdot \text{sign}(\nabla f(\mathbf{x}_0))$, which reduces to FGSM [4] when $\gamma_t = 1$. Therefore, our proposed Frank-Wolfe white-box attack also includes FGSM as a one-step special instance.

[3] The derivation can be found in the Appendix.

[4] The extra clipping operation in FGSM is to project to the additional box constraint for image classification task. We will also need this clipping operation at the end of each iteration for specific tasks such as image classification.

2.3.5 Frank-Wolfe Black-box Attacks

Next we consider the black-box setting, where we cannot perform back-propagation to calculate the gradient of the loss function anymore. Instead, we can only query the DNN system's outputs with specific inputs. To clarify, here the output refers to the logit layer's output (confidence scores for classification), not the final prediction label.

We propose a zeroth-order Frank-Wolfe based algorithm to solve this problem in Algorithm 2. The key difference between our proposed black-box attack and white-box attack is one extra gradient estimation step, which is presented in Line 4 in Algorithm 2. Also, the momentum term $\mathbf{m}_t$ is now defined as the exponential average of previous gradient estimations $\{\mathbf{q}_t\}_{t=0}^{T-1}$. This will help reduce the variance in zeroth-order gradient estimation and empirically accelerate the convergence of Algorithm 2.

Algorithm 2 Frank-Wolfe Black-box Attack Algorithm

```
 1: input: number of iterations $T$, step sizes $\{\gamma_t\}$, sample size for gradient estimation $b$,
    sampling parameter $\delta$;
 2: $\mathbf{x}_0 = \mathbf{x}_{\text{ori}}$, $\mathbf{m}_{-1} = \text{GRAD\_EST}(\mathbf{x}_0, b, \delta)$
 3: for $t = 0, \ldots, T-1$ do
 4:     $\mathbf{q}_t = \text{GRAD\_EST}(\mathbf{x}_t, b, \delta)$  // Alg 3
 5:     $\mathbf{m}_t = \beta \cdot \mathbf{m}_{t-1} + (1-\beta) \cdot \mathbf{q}_t$
 6:     $\mathbf{v}_t = \text{argmin}_{\mathbf{v} \in \mathcal{X}} \langle \mathbf{v}, \mathbf{m}_t \rangle$
 7:     $\mathbf{d}_t = \mathbf{v}_t - \mathbf{x}_t$
 8:     $\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t \mathbf{d}_t$
 9: end for
10: output: $\mathbf{x}_T$
```

As in many other zeroth-order optimization algorithms [Sha17, FKM05], Algorithm 3 uses symmetric finite differences to estimate the gradient and therefore, gets rid of the dependence on back-propagation in white-box setting. Different from [CZS$^+$17], here we do not utilize natural basis as our sensing vectors, instead, we provide two options: one is to use vectors uniformly sampled from Euclidean unit sphere and the other is to use vectors uniformly sampled from standard multivariate Gaussian distribution. This will greatly improve the gradient estimation efficiency comparing to sensing with natural bases as such option will only be able to estimate one coordinate of the gradient vector per query. In practice, both options here provide us competitive experimental results. It is worth noting that NES method [WSG+14] with antithetic sampling [SHC+17] used in [IEA+18] yields similar formula as our option II in Algorithm 3.

Algorithm 3 GRAD_EST($\mathbf{x}, b, \delta$)

```
1: $\mathbf{q} = \mathbf{0}$
2: for $i = 1, \ldots, b$ do
3:     option I: Sample $\mathbf{u}_i$ uniformly from the Euclidean unit sphere with $\|\mathbf{u}_i\|_2 = 1$
           $\mathbf{q} = \mathbf{q} + \frac{d}{2\delta b}\big(f(\mathbf{x} + \delta\mathbf{u}_i) - f(\mathbf{x} - \delta\mathbf{u}_i)\big)\mathbf{u}_i$
4:     option II: Sample $\mathbf{u}_i$ uniformly from the standard Gaussian distribution $N(\mathbf{0}, \mathbf{I})$
           $\mathbf{q} = \mathbf{q} + \frac{1}{2\delta b}\big(f(\mathbf{x} + \delta\mathbf{u}_i) - f(\mathbf{x} - \delta\mathbf{u}_i)\big)\mathbf{u}_i$
5: end for
6: return $\mathbf{q}$
```

2.4 Main Theory

In this section, we establish the convergence guarantees for our proposed Frank-Wolfe adversarial attack algorithms described in Section 3.3. The omitted proofs can be found in the Appendix. First, we introduce the convergence criterion for our Frank-Wolfe adversarial attack framework.

2.4.1 Convergence Criterion

The loss function for common DNN models are generally nonconvex. In addition, (2.3.1) is a constrained optimization. For such general nonconvex constrained optimization, we typically adopt the Frank-Wolfe gap as the convergence criterion (since gradient norm of f is no longer a proper criterion for constrained optimization problems):

$$g(\mathbf{x}_t) = \max_{\mathbf{x} \in \mathcal{X}} \langle \mathbf{x} - \mathbf{x}_t, -\nabla f(\mathbf{x}_t) \rangle.$$

Note that we always have $g(\mathbf{x}_t) \geq 0$ and $\mathbf{x}_t$ is a stationary point for the constrained optimization problem if and only if $g(\mathbf{x}_t) = 0$, which makes $g(\mathbf{x}_t)$ a perfect convergence criterion for Frank-Wolfe based algorithms.

2.4.2 Convergence Guarantee for Frank-Wolfe White-box Attack

Before we are going to provide the convergence guarantee of the Frank-Wolfe white-box attack (Algorithm 1), we introduce the following assumptions that are essential to the convergence analysis.

Assumption 2.4.1. Function $f(\cdot)$ is L-smooth with respect to $\mathbf{x}$, i.e., for any $\mathbf{x}, \mathbf{x}'$, it holds that

$$f(\mathbf{x}') \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{x}' - \mathbf{x}) + \frac{L}{2} \|\mathbf{x}' - \mathbf{x}\|_2^2.$$

Assumption 2.4.1 is a standard assumption in nonconvex optimization, and is also adopted in other Frank-Wolfe literature such as [LJ16, RSPS16]. Note that even though the smoothness assumption does not hold for general DNN models, a recent study [STIM18] shows that batch normalization that is used in many modern DNNs such as the Inception V3 model, actually makes the optimization landscape significantly smoother [5]. In addition, recent studies

[5] The original argument in [STIM18] refers to the smoothness with respect to each layer's parameters. Note that the first layer's parameters are in the mirror position (in terms of backpropagation) as the network inputs. Therefore, the argument in [STIM18] can also be applied here with respect to the network inputs.

[AZLS19, DLL$^+$19, ZCZG19] also showed that the loss function of overparameterized deep neural networks is semi-smooth. This justifies the validity of Assumption 2.4.1.

Assumption 2.4.2. Set $\mathcal{X}$ is bounded with diameter D, i.e., $\|\mathbf{x}-\mathbf{x}'\|_2 \leq D$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

Assumption 2.4.2 implies that the input space is bounded. For common tasks such as image classification, given the fact that images have bounded pixel range and ϵ is a small constant, this assumption trivially holds. Given the above assumptions, the following lemma shows that the momentum term $\mathbf{m}_t$ will not deviate from the gradient direction significantly.

Lemma 2.4.3. Under Assumptions 2.4.1 and 2.4.2, for $\mathbf{m}_t$ in Algorithm 1, it holds that

$$\|\nabla f(\mathbf{x}_t) - \mathbf{m}_t\|_2 \leq \frac{\gamma\beta LD}{1-\beta}.$$

Now we present the theorem, which characterizes the convergence rate of our proposed Frank-Wolfe white-box adversarial attack algorithm presented in Algorithm 1.

Theorem 2.4.4. Under Assumptions 2.4.1 and 2.4.2, let
$\gamma_t = \gamma = \sqrt{2(f(\mathbf{x}_0) - f(\mathbf{x}^*))/(C_\beta LD^2T)}$, the output of Algorithm 1 satisfies

$$\widetilde{g}_T \leq \sqrt{\frac{2C_\beta LD^2(f(\mathbf{x}_0) - f(\mathbf{x}^*))}{T}},$$

where $\widetilde{g}_T = \min_{1\leq k\leq T} g(\mathbf{x}_k)$, $\mathbf{x}^*$ is the optimal solution to (2.3.1) and $C_\beta = (1+\beta)/(1-\beta)$.

Remark 2.4.5. Theorem 2.4.4 suggests that our proposed Frank-Wolfe white-box attack algorithm achieves a $O(1/\sqrt{T})$ rate of convergence. Unlike previous work [LJ16] which focuses on the convergence rate of the classic Frank-Wolfe method, our analysis shows the convergence rate of the Frank-Wolfe method with momentum mechanism.

2.4.3 Convergence Guarantee for Frank-Wolfe Black-box Attack

Next we analyze the convergence of our proposed Frank-Wolfe black-box adversarial attack algorithm presented in Algorithm 2.

In order to prove the convergence of our proposed Frank-Wolfe black-box attack algorithm, we need the following additional assumption that $\|\nabla f(\mathbf{0})\|_2$ is bounded.

Assumption 2.4.6. Gradient of $f(\cdot)$ at zero point $\nabla f(\mathbf{0})$ satisfies $\max_y \|\nabla f(\mathbf{0})\|_2 \le G$.

Following the analysis in [Sha17], let $f_\delta(\mathbf{x}) = \mathbb{E}_{\mathbf{u}}[f(\mathbf{x}+\delta\mathbf{u})]$, which is the smoothed version of $f(\mathbf{x})$. This smoothed function value plays a central role in our theoretical analysis, since it bridges the finite difference gradient approximation with the actual gradient. The following lemma shows this relationship.

Lemma 2.4.7. For any $\mathbf{x}$ and the gradient estimator $\mathbf{q}$ of $\nabla f(\mathbf{x})$ in Algorithm 3, its expectation and variance satisfy

$$\mathbb{E}[\mathbf{q}] = \nabla f_\delta(\mathbf{x}),\ \mathbb{E}\|\mathbf{q} - \mathbb{E}[\mathbf{q}]\|_2^2 \le \frac{1}{b}\bigg(2d(G+LD)^2 + \frac{1}{2}\delta^2 L^2 d^2\bigg).$$

And also we have

$$\mathbb{E}\|\nabla f(\mathbf{x}) - \mathbf{q}\|_2 \le \frac{\delta L d}{2} + \frac{2\sqrt{d}(G+LD) + \delta L d}{\sqrt{2b}}.$$

Now we are going to present the theorem, which characterizes the convergence rate of Algorithm 2.

Theorem 2.4.8. Under Assumptions 2.4.1, 2.4.2 and 2.4.6, let
$\gamma_t = \gamma = \sqrt{(f(\mathbf{x}_0) - f(\mathbf{x}^*))/(C_\beta L D^2 T)}$, $b = Td$ and $\delta = \sqrt{1/(Td^2)}$, the output of Algorithm 2 satisfies

$$\mathbb{E}[\widetilde{g}_T] \le \frac{D}{\sqrt{T}}\Big(\sqrt{2C_\beta L(f(\mathbf{x}_0) - f(\mathbf{x}^*))} + C_\beta(L + G + LD)\Big),$$

where $\widetilde{g}_T = \min_{1\le k\le T} g(\mathbf{x}_k)$, the expectation of $\widetilde{g}_T$ is over the randomness of the gradient estimator, $\mathbf{x}^*$ is the optimal solution to (2.3.1) and $C_\beta = (1+\beta)/(1-\beta)$.

Remark 2.4.9. Theorem 2.4.8 suggests that Algorithm 2 also enjoys a $O(1/\sqrt{T})$ rate of convergence. Note that [BG18] proves the convergence rate for the classic zeroth-order Frank-Wolfe algorithm. Our result is different in several aspects. First, we prove the convergence rate of zeroth-order Frank-Wolfe with momentum. Second, we use the symmetric finite difference gradient estimator with two types of sensing vectors while they [BG18] use one-side finite difference gradient estimator with Gaussian sensing vectors. In terms of query complexity, the total number of queries needed in Algorithm 2 is $Tb = T^2 d$, which is linear in the data dimension d. In fact, in the experiment part, we observe that this number can be substantially smaller than d, e.g., $b = 25$.

2.5 Experiments

In this section, we present the experimental results for our proposed Frank-Wolfe attack framework against other state-of-the-art adversarial attack algorithms in both white-box and black-box settings. All of our experiments are conducted on Amazon AWS p3.2xlarge servers which come with Intel Xeon E5 CPU and one NVIDIA Tesla V100 GPU (16G RAM). All experiments are implemented in Tensorflow platform version 1.10.0 within Python 3.6.4.

2.5.1 Evaluation Setup

We compare the performance of all attack algorithms by evaluating on both MNIST [LeC98] and ImageNet [DDS+09] datasets. For the MNIST dataset, we attack a pre-trained 6-layer CNN: 4 convolutional layers followed by 2 dense layers with max-pooling and Relu activations applied after each convolutional layer. The pre-trained model achieves 99.3% accuracy on the MNIST test set. For ImageNet experiments, we attack a pre-trained Inception V3 model [SVI+16]. The pre-trained Inception V3 model is reported to have a 78.0% top-1 accuracy and a 93.9% top-5 accuracy. For the MNIST dataset, we randomly choose 1000 images from its test set that are verified to be correctly classified by the pre-trained model and also randomly choose a target class for each image. Similarly, for the ImageNet dataset, we randomly choose 250 images from its validation set as our attack examples. For our proposed black-box attack, we test both options in Algorithm 3. We performed a grid search to tune the hyper-parameters for all algorithms to ensure a fair comparison. A detailed description of hyperparameter tuning and parameter settings can be found in the Appendix.

2.5.2 Baseline Methods

We compare the proposed algorithms with several state-of-the-art baseline algorithms. Specifically, we compare the proposed white-box attack algorithm with (i) FGSM [GSS15] (ii) PGD [MMS+18] (normalized steepest descent[6]) (iii) MI-FGSM [DLP+18]. We compare the proposed black-box attack algorithm with (i) NES-PGD attack [IEA+18] and (ii) Bandit attack [IEM19]. We did not report the comparison with ZOO [CZS+17] here because it consistently underperforms NES-PGD and Bandit attacks according to our experiments and prior work. We also compare with [LLW+19] on attacking the robust model trained by adversarial training.

2.5.3 White-box Attack Experiments

In this subsection, we present the white-box attack experiments on both MNIST and ImageNet datasets. We choose $\epsilon = 0.3$ for MNIST dataset and $\epsilon = 0.05$ for ImageNet dataset. For comparison, we report the attack success rate, the average number of iterations to complete the attack, as well as average distortion for each method.

Tables 2.1 and 2.2 present our experimental results for the white-box attack experiments. For experiments on both datasets, while FGSM only needs 1 gradient update per attack, it only achieves 21.5% attack success rate on MNIST and 1.2% attack success rate on ImageNet in the targeted attack setting. All the other methods achieve 100% attack success rate. PGD needs on average 6.2 and 8.7 gradient iterations per attack on MNIST and ImageNet respectively. MI-FGSM improves it to around 4.0 and 5.0 iterations per attack on MNIST

[6]standard PGD will need large step size to go anywhere since the gradient around the true example is relatively small. On the other hand, the large step size will cause the algorithm go out of the constraint set quickly and basically stop moving since then because of the projection step.

and ImageNet. However, the distortion of both PGD and MI-FGSM is very close to the perturbation limit ϵ, which indicates that their generated adversarial examples are near or upon the boundary of the constraint set. On the other hand, our proposed Frank-Wolfe white-box attack algorithm achieves not only the smallest average number of iterations per attack, but also the smallest distortion among the baselines. This suggests the advantage of Frank-Wolfe based projection-free algorithms for the white-box attack.

Table 2.1: Comparison of targeted L_∞ norm based white-box attacks on MNIST dataset with $\epsilon = 0.3$.

Methods	ASR(%)	# Iterations	Distortion
FGSM	21.5	-	0.300
PGD	100.0	6.2	0.277
MI-FGSM	100.0	4.0	0.279
FW-white	100.0	**3.3**	**0.256**

Table 2.2: Comparison of targeted L_∞ norm based white-box attacks on ImageNet dataset with $\epsilon = 0.05$.

Methods	ASR(%)	# Iterations	Distortion
FGSM	1.2	-	0.050
PGD	100.0	8.7	0.049
MI-FGSM	100.0	5.0	0.049
FW-white	100.0	**4.8**	**0.019**

Table 2.3: Comparison of targeted L_∞ norm based black-box attacks on MNIST and ImageNet datasets in terms of attack success rate, average time and the average number of queries (QUERIES: for all images including both successfully and unsuccessfully attacked ones; QUERIES(SUCC): for successfully attacked ones only) needed per image.

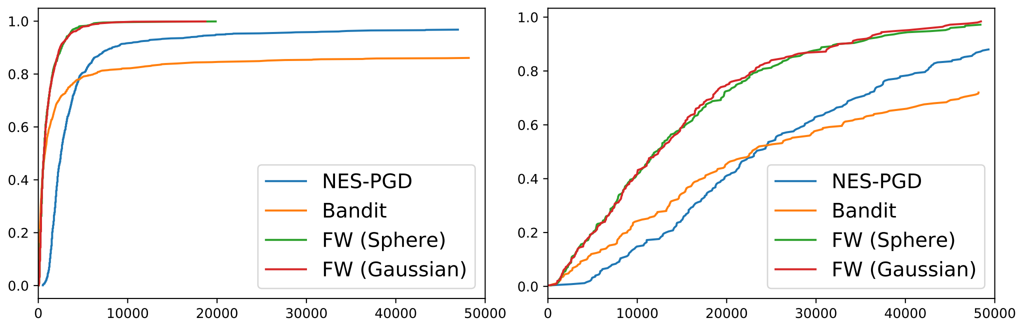
Methods	MNIST ($\epsilon = 0.3$)				ImageNet ($\epsilon = 0.05$)			
	ASR(%)	Time(s)	Queries	Queries(succ)	ASR(%)	Time(s)	Queries	Queries(succ)
NES-PGD	96.8	0.2	5349.0	3871.3	88.0	85.1	26302.8	23064.5
Bandit	86.1	4.8	8688.9	2019.7	72.0	148.7	27172.5	18295.2
FW (Sphere)	**99.9**	**0.1**	**1132.6**	**1083.6**	97.2	62.1	15424.0	**14430.8**
FW (Gaussian)	**99.9**	**0.1**	1144.4	1095.4	**98.4**	**50.6**	**15099.4**	14532.3

2.5.4 Black-box Attack Experiments

In this subsection, we present the black-box attack experiments on both MNIST and ImageNet datasets. The maximum query limit is set to be $50,000$ per attack. We choose $\epsilon = 0.3$ for MNIST dataset and $\epsilon = 0.05$ for ImageNet dataset. For comparison, we report the attack success rate, average attack time, average number of queries needed, as well as average number of queries needed on successfully attacked samples for each method.

Table 2.3 presents our experimental results for targeted black-box attacks on both ImageNet and MNIST datasets. We can see that on MNIST, NES-PGD method achieves a relatively high attack success rate, but still takes quite a lot of queries per (successful) attack. Bandit method improves the query complexity for successfully attacked samples but has a lower attack success rate in this setting and takes a longer time to complete the attack. In sharp contrast, our proposed Frank-Wolfe black-box attack algorithms (both sphere and Gaussian sensing vector options) achieve the highest success rate in the targeted black-box attack setting while greatly improve the query complexity by around 50% over the best baseline. On ImageNet, similar patterns can be observed: our proposed Frank-Wolfe black-box attack algorithms achieve the highest attack success rate and further significantly improve the query efficiency against the baselines. This suggests the advantage of Frank-Wolfe based projection-free algorithms for the black-box attack.

To provide more intuitive demonstrations, we also plot the attack success rate against the number of queries for our black-box experiments. Figure 2.1 shows the plots of the attack success rate against the number of queries for different algorithms on MNIST and ImageNet datasets respectively. As we can see from the plots, The bandit attack achieves better query efficiency for easy-to-attack examples (require fewer queries to attack) compared with NES-PGD or even FW at the early stages, but falls behind even to NES-PGD on hard-to-attack examples (require more queries to attack). We conjecture that in the targeted attack setting, the gradient/data priors are not as accurate as in the untargeted attack setting, which makes the Bandit attack less effective especially on hard-to-attack examples. On the other hand, our proposed Frank-Wolfe black-box attack algorithms achieve the highest attack success rate and the best efficiency (least queries needed for achieving the same success rate). This again confirms the advantage of Frank-Wolfe based projection-free algorithms for the black-box attack.



(a) MNIST (b) ImageNet

Figure 2.1: Attack success rate against the number of queries plot for targeted black-box attacks on MNIST and ImageNet datasets.

2.5.5 Experiments on Adversarially Trained Model

In this subsection, we further present the white-box and black-box attack experiments on the more challenging robust CIFAR10 model. Specifically, we apply the proposed Frank-Wolfe white-box and black-box attack algorithms to adversarially trained WideResNet model using adversarial training [MMS+18]. Following [MMS+18], we choose $\epsilon = 8/255$. For the black-box case, the maximum query limit is set to be $20,000$ per attack. Table 2.4 presents our experimental results for targeted white-box attacks on robust CIFAR10 model. Specifically, in the white-box case, the proposed Frank-Wolfe attack achieves 24.3% attack success rate [7] with the smallest L_∞ distortion, while PGD and MI-FGSM can only achieve lower attack success rates and also larger distortions. Table 2.5 presents our experimental results for targeted black-box attacks on robust CIFAR10 model. In black-box setting, our algorithm achieves 19.0% attack success rate with the smallest overall queries (also a relatively small number of queries for successful attempts) while NES needs a larger number of queries but achieves only 9.4% attack success rate. Bandit improves the number of average queries needed for successful attempts, yet its attack success rate is only 9.6%. The Nattack achieves an attack success rate slightly better than Frank-Wolfe but requires the largest number of queries for successful attempts.

2.6 Conclusions and Future Work

In this work, we propose a Frank-Wolfe framework for efficient and effective adversarial attacks. Our proposed white-box and black-box attack algorithms enjoy an $O(1/\sqrt{T})$ rate of convergence, and the query complexity of the proposed black-box attack algorithm is linear in data dimension d. Finally, our empirical study on attacking both the ImageNet dataset and the MNIST dataset yields the best distortion in the white-box setting and the highest attack success rate/query complexity in the black-box setting.

[7] note that it is the targeted attack, so the number is much lower than the original paper of [MMS+18]

Table 2.4: Comparison of targeted L_∞ norm based while-box attacks on adversarially trained WideResNet on CIFAR10 with $\epsilon = 8/255$.

Methods	ASR(%)	# Iterations	Distortion
FGSM	21.5	-	8.00
PGD	24.0	**15.6**	7.49
MI-FGSM	24.1	15.8	7.60
FW-white	**24.3**	15.8	**7.48**

Table 2.5: Comparison of targeted L_∞ norm based black-box attacks on adversarially trained WideResNet on CIFAR10 with $\epsilon = 8/255$ in terms of attack success rate and the average number of queries (QUERIES: for all images including both successfully and unsuccessfully attacked ones; QUERIES(SUCC): for successfully attacked ones only) needed per image.

Methods	ASR(%)	# Queries	Queries(SUCC)
NES-PGD	9.4	18541.1	4480.1
Bandit	9.6	18174.2	**981.5**
Nattack	**20.0**	17135.0	5675.0
FW (Opt I)	19.0	**16735.2**	2816.8
FW (Opt II)	16.8	16748.2	2703.2

It would also be interesting to see whether the performance of our Frank-Wolfe adversarial framework can be further improved by incorporating the idea of gradient/data priors [IEM19]. We leave it as future work.

CHAPTER 3

RayS: A Ray Searching Method for Hard-label Adversarial Attack

3.1 Introduction

Deep neural networks (DNNs) have achieved remarkable success on many machine learning tasks such as computer vision [HZRS16a, SHK12], and speech recognition [HDY+12] in the last decade. Despite the great success, recent studies have shown that DNNs are vulnerable to adversarial examples, i.e., even imperceptible (specially designed not random) perturbations could cause the state-of-the-art classifiers to make wrong predictions [SZS+13, GSS15]. This intriguing phenomenon has soon led to an arms race between adversarial attacks [CW17, ACW18, CZYG20] that are trying to break the DNN models with such small perturbations and adversarial defenses methods [PMW+16, MMS+18, WMB+19, ZYJ+19, WZY+20] that tries to defend against existing attacks. During this arm race, many heuristic defenses [PMW+16, GRCVDM18, XWZ+18, SKN+18, MLW+18, SKC18, DAL+18] are later proved to be not effective under harder attacks. One exception is adversarial training [GSS15, MMS+18], which was demonstrated as an effective defense approach.

A large body of adversarial attacks has been proposed during this arm race. According to the different amounts of information the attacker could access, adversarial attacks can be generally divided into three categories: white-box attacks, black-box attacks, and hard-label attacks. White-box attacks [MMS+18, CW17] refer to the case where the attacker has access to all information regarding the target model, including the model weights, structures, parameters, and possible defense mechanisms. Since white-box attackers could access all model details, they can efficiently perform back-propagation on the target model and compute gradients. In black-box attacks, the attacker only has access to the queried soft label output (logits or probability distribution of different classes) of the target model, and the other parts are treated as a black-box. The black-box setting is much more practical compared with the white-box case, however, in such a setting, the attacker cannot perform back-propagation and direct gradient computation. Therefore, many turn to transfer the gradient from a known model [PMG16] or estimate the true gradient via zeroth-order optimization methods [IEA+18, IEM19, CZYG20, ADO20].

Hard-label attacks, also known as decision-based attacks, on the other hand, only allow the attacker to query the target model and get hard-label output (prediction label). Obviously, the hard-label setting is the most challenging one, yet it is also the most practical one, as in reality, there is little chance that the attacker could know all the information about the target model in advance or get the probability prediction of all classes. The hard-label-only access also means that the attacker cannot tell the subtle changes in the target model's output when feeding a slightly perturbed input sample (assuming this slight perturbation will not change the model prediction). Therefore, the attacker can only find informative clues around the decision boundary of the target model where tiny perturbations could cause the model to have different prediction labels. Previous works [BRB18, CLC+19, CSC+20, CJW19] mostly follow this idea to tackle the hard-label adversarial attack problem. However, [BRB18, CLC+19, CSC+20, CJW19] are all originally proposed for L_2 norm threat model while L_∞ norm threat models [MMS+18, ZYJ+19, KW20, ZW19, ZX20] are currently the most popular and widely used. Even though [CLC+19, CSC+20, CJW19] provide extensions to L_∞ norm case, none of them has been optimized for the L_∞ norm case and consequently, their attack performance falls largely behind traditional L_∞ norm based white-box and black-box attacks, making them inapplicable in real world scenarios. This leads to a natural question that, *Can we design a hard-label attack that could greatly improve upon previous hard-label attacks and provide practical attacks for the most widely used L_∞ norm threat model?*

In this paper, we answer this question affirmatively. We summarize our main contributions as follows

- We propose the Ray Searching attack, which only relies on the hard-label output of the target model. We show that the proposed hard-label attack is much more effective and efficient than previous hard-label attacks in the L_∞ norm threat model.
- Unlike previous works, most of which solve the hard-label attack problem via zeroth-order optimization methods, we reformulate the continuous optimization problem of finding the closest decision boundary into a discrete one and directly search for the closest decision boundary along a discrete set of ray directions. A fast check step is also utilized to skip unnecessary searches. This significantly saves the number of queries needed for the hard-label attack. Our proposed attack is also free of hyperparameter tuning such as step size or finite difference constant, making itself very stable and easy to apply.
- Moreover, our proposed RayS attack can also be used as a strong attack to detect possible "falsely robust" models. By evaluating several recently proposed defenses that claim to achieve the state-of-the-art robust accuracy with RayS attack, we show that the current white-box/black-box attacks can be deceived and give a false sense of security. Specifically, the RayS attack significantly decreases the robust accuracy of the most popular PGD attack on several robust models and the difference could be as large as 28%. We believe that our proposed RayS attack could help identify falsely robust models that deceive current white-box/black-box attacks.

Notation. For a d-dimensional vector $\mathbf{x} = [x_1, ..., x_d]^\top$, we use $\|\mathbf{x}\|_0 = \sum_i \mathbb{1}\{x_i \neq 0\}$ to denote its ℓ_0-norm, use $\|\mathbf{x}\|_2 = (\sum_{i=1}^d |x_i|^2)^{1/2}$ to denote its ℓ_2-norm and use $\|\mathbf{x}\|_\infty = \max_i |x_i|$

to denote its ℓ_∞-norm, where $\mathbb{1}(\cdot)$ denotes the indicator function.

3.2 Related Work

There is a large body of works on evaluating model robustness and generating adversarial examples. In this section, we review the most relevant works with ours.

White-box attacks: [SZS+13] first brought up the concept of adversarial examples and adopt the L-BFGS algorithm for attacks. [GSS15] proposed the Fast Gradient Sign Method (FGSM) method via linearizing the network loss function. [KGB16] proposed to iteratively perform FGSM and conduct projection afterward, which is equivalent to Projected Gradient Descent (PGD) [MMS+18]. [PMJ+16] proposed JSMA method based on the Jacobian saliency map and [MDFF16] proposed DeepFool attack by projecting the data to the closest separating hyper-plane. [CW17] introduced the CW attack with a margin-based loss function and show that defensive distillation [PMW+16] is not truly robust. [CZYG20] proposed a projection-free attack based on the Frank-Wolfe method with momentum. [ACW18] identified the effect of obfuscated gradients and proposed the BPDA attack for breaking those obfuscated gradient defenses.

Black-box attacks: Other than the aforementioned white-box attack algorithms, there also exists a large body of literature [HT17, PMG16, PMG+17, CZS+17, IEA+18, IEM19, LLW+19, CZYG20] focusing on the black-box attack case where the information is limited to the logits output of the model rather than every detail of the model. Transfer-based black-box attacks [HT17, PMG16, PMG+17] try to transfer the gradient from a known model to the black-box target model and then apply the same technique as in the white-box case. However, their attack effectiveness is often not quite satisfactory. Optimization-based black-box attacks aim to estimate the true gradient via zeroth-order optimization methods. [CZS+17] proposed to estimate the gradient via finite-difference on each dimension. [IEA+18] proposed to improve the query efficiency of [CZS+17] via Natural Evolutionary Strategies.

[IEM19] further improved upon [IEA$^+$18] by exploiting gradient priors. [UOKO18] proposed to use the SPSA method to build a gradient-free attack that can break vanishing gradient defenses. [ADO20] proposed to directly estimate the sign of the gradient instead of the true gradient itself. [MAS19] reformulated the continuous optimization problem into a discrete one and proposed a combinatorial search based algorithm to make the attack more efficient. [ACFH19] proposed a randomized search scheme to iteratively patch small squares onto the test example.

Hard-label attacks: [BRB18] first studied the hard-label attack problem and proposed to solve it via random walks near the decision boundary. [IEA$^+$18] demonstrated a way to transform the hard-label attack problem into a soft label attack problem. [CLC$^+$19] turned the adversarial optimization problem into the problem of finding the optimal direction that leads to the shortest L_2 distance to decision boundary and optimized the new problem via zeroth-order optimization methods. [CSC$^+$20] further improved the query complexity of [CLC$^+$19] by estimating the sign of gradient instead of the true gradient. [CJW19] also applied zeroth-order sign oracle to improve [BRB18] by searching the step size and keeping the iterates along the decision boundary.

3.3 The Proposed Method

In this section, we introduce our proposed *Ray Searching attack* (RayS). Before we go into details about our proposed method, we first take an overview of the previous adversarial attack problem formulations.

3.3.1 Overview of Previous Problem Formulations

We denote the DNN model by f and the test data example as $\{\mathbf{x}, y\}$. The goal of adversarial attack is to solve the following optimization problem

$$\min_{\mathbf{x}'} \mathbb{1}\{f(\mathbf{x}') = y\} \quad \text{s.t.,} \quad \|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon, \tag{3.3.1}$$

where ϵ denotes the maximum allowed perturbation strength. The indicator function $\mathbb{1}\{f(\mathbf{x}') = y\}$ is hard to optimize, therefore, [MMS+18, ZYJ+19, CZYG20, IEA+18, IEM19, ADO20] turn to relax (3.3.1) into

$$\max_{\mathbf{x}'} \ell(f(\mathbf{x}'), y) \quad \text{s.t.,} \quad \|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon, \tag{3.3.2}$$

where ℓ denotes the surrogate loss function such as CrossEntropy loss. On the other hand, traditional hard-label attacks [CLC+19, CSC+20] re-formulate (3.3.1) as

$$\min_{\mathbf{d}} g(\mathbf{d}) \quad \text{where} \quad g(\mathbf{d}) = \operatorname*{argmin}_{r} \mathbb{1}\{f(\mathbf{x} + r\mathbf{d}/\|\mathbf{d}\|_2) = y\}. \tag{3.3.3}$$

Here $g(\mathbf{d})$ represents the decision boundary radius from original example $\mathbf{x}$ along ray direction $\mathbf{d}$ and the goal is to find the minimum decision boundary radius regarding the original example $\mathbf{x}$. Let $(\widehat{r}, \widehat{\mathbf{d}})$ denotes the minimum decision boundary radius and the corresponding ray direction. If the minimum decision boundary radius satisfies $\|\widehat{r}\widehat{\mathbf{d}}/\|\widehat{\mathbf{d}}\|_2\|_\infty \leq \epsilon$, it will be counted as a successful attack.

While prior works [CLC+19, CSC+20] try to solve problem (3.3.3) in a continuous fashion by estimating the gradient of $g(\mathbf{d})$ via zeroth-order optimization methods, the hard-label-only access restriction imposes great challenges in solving (3.3.3). Specifically, estimating the the decision boundary radius $g(\mathbf{d})$ typically takes a binary search procedure and estimating an informative gradient of $g(\mathbf{d})$ via finite difference requires multiple rounds of $g(\mathbf{d})$ computation. Furthermore, due to the large variance in zeroth-order gradient estimating procedure, optimizing (3.3.3) typically takes a large number of gradient steps. These together, make solving (3.3.3) much less efficient and effective than black-box attacks, not to mention white-box attacks.

Given all the problems mentioned above, we turn to directly search for the closest decision boundary without estimating any gradients.

3.3.2 Ray Search Directions

With a finite number of queries, it is impossible to search through the whole continuous ray direction space. As a consequence, we need to restrict the search space to a discrete set of ray directions to make direct searches possible. Note that applying FGSM to (3.3.2) leads to an optimal solution at the vertex of the L_∞ norm ball [MAS19, CZYG20], suggesting that those vertices might provide possible solutions to (3.3.2). Empirical findings in [MAS19] also suggest that the solution to (3.3.2) obtained from the PGD attack is mostly found on the vertices of L_∞ norm ball. Inspired by this, [MAS19] restrict the feasible solution set as the vertex of the L_∞ norm ball. Following this idea, since our goal is to obtain the decision boundary radius, we consider the ray directions that point to the L_∞ norm ball vertices, i.e., $\mathbf{d} \in \{-1, 1\}^d$ where d denotes the dimension of original data example $\mathbf{x}$[1]. Therefore, instead of solving (3.3.3), we turn to solve a discrete problem

$$\min_{\mathbf{d} \in \{-1,1\}^d} g(\mathbf{d}) \quad \text{where} \quad g(\mathbf{d}) = \operatorname*{argmin}_r \mathbb{1}\{f(\mathbf{x} + r\mathbf{d}/\|\mathbf{d}\|_2) = y\}. \tag{3.3.4}$$

In problem (3.3.4), we reduce the search space from $\mathbb{R}^d$ to $\{-1, 1\}^d$, which contains 2^d possible search directions.

Now we begin to introduce our proposed Ray Searching attack. We first present the naive version of the Ray Searching attack, which is summarized in Algorithm 4. Specifically, given a model f and a test example $\{\mathbf{x}, y\}$, we first initialize the best search direction as an all-one vector and set the initial best radius as infinity. Then we iteratively change the sign of each dimension of the current best ray direction and test whether this modified ray direction leads to a better decision boundary radius by Algorithm 5 (will be described later). If it does, we update the best search direction and the best radius, otherwise, they remain unchanged.

[1]Without loss of generality, here we view $\mathbf{x}$ simply as a d-dimensional vector.

Algorithm 4 is a greedy search algorithm that finds the local optima of the decision boundary radius, where the local optima of the decision boundary radius are defined as follows.

Definition 3.3.1 (Local Optima of Decision Boundary Radius)**.** A ray direction $\mathbf{d} \in \{-1, 1\}^d$ is the local optima of the decision boundary radius regarding (3.3.4), if for all $\mathbf{d}' \in \{-1, 1\}^d$ satisfy $\|\mathbf{d}' - \mathbf{d}\|_0 \leq 1$, we have $g(\mathbf{d}) \leq g(\mathbf{d}')$.

Theorem 3.3.2. Given enough query budgets, let $(\widehat{r}, \widehat{\mathbf{d}})$ be the output of Algorithm 4, then $\widehat{\mathbf{d}}$ is the local optima of decision boundary radius problem (3.3.4).

Proof. We prove this by contradiction. Suppose $\widehat{\mathbf{d}}$ is not the local optima, there must exist some $\mathbf{d}'$ satisfying $\|\mathbf{d}' - \widehat{\mathbf{d}}\|_0 \leq 1$, i.e., $\mathbf{d}'$ differs from $\widehat{\mathbf{d}}$ by at most 1 dimension, that $g(\widehat{\mathbf{d}}) > g(\mathbf{d}')$. This means Algorithm 4 can still find better solution than $g(\widehat{\mathbf{d}})$ by going through all dimensions and thus $\widehat{\mathbf{d}}$ will not be the output of Algorithm 4. This leads to a contradiction. $\square$

Next we introduce Algorithm 5, which performs decision boundary radius search. The main body of Algorithm 5 (from Line 7 to Line 12) is a binary search algorithm to locate the decision boundary radius with high precision. The steps before Line 7, on the other hand, focus on deciding the search range and whether we need to search it (this is the key to achieve efficient attacks). Specifically, we first normalize the search direction by its L_2 norm. And then in Line 3, we do a fast check at $x + r_{\text{best}} \cdot \mathbf{d}_n$[2] and decide whether we need to further perform a binary search for this direction. To help better understand the underlying mechanism, Figure 3.1 provides a two-dimensional sketch for the fast check step in Line 3 in Algorithm 5. Suppose we first change the sign of the current $\mathbf{d}_{\text{best}}$ at dimension 1, resulting a modified direction $\mathbf{d}_{\text{tmp1}}$. The fast check shows that it is a valid attack and it has the potential to further reduce the decision boundary radius. On the other hand, if we change the sign of $\mathbf{d}_{\text{best}}$ at dimension 2, resulting a modified direction $\mathbf{d}_{\text{tmp2}}$. The fast check shows

[2]For applications such as image classification, there is an additional clipping to $[0, 1]$ operation to keep the image valid. We assume this is included in model f and do not write it explicitly in Algorithm 5.

that it is no longer a valid attack and the decision boundary radius of direction $\mathbf{d}_{\text{tmp2}}$ can only be worse than the current r_{best}. Therefore, we skip all unnecessary queries that aim to estimate a worse decision boundary radius. Note that in [CSC+20], a similar check was also presented for slightly perturbed directions. However, they use it as the sign for gradient estimation while we simply drop all unsatisfied radius based on the check result and obtain better efficiency. Finally, we explain Line 6 in Algorithm 5. The choice of $\min(r_{\text{best}}, \|\mathbf{d}\|_2)$ is because initial r_{best} is ∞, in the case where the fast check passes, we should make sure the binary search range is finite.

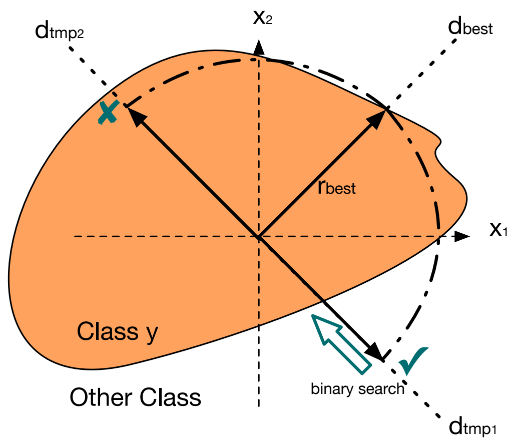


Figure 3.1: A two-dimensional sketch for the fast check step in Algorithm 5.

3.3.3 Hierarchical Search

Recent works on black-box attacks [IEM19, MAS19] found that there exists some spatial correlation between different dimensions of the gradients, and exploiting this prior could help improve the efficiency of black-box attacks. Therefore, they added the same perturbation for small tiles or image blocks on the original data example to achieve better efficiency. Inspired by this finding, we also exploit these spatial correlations by designing a hierarchical search version of the Ray Searching attack, displayed in Algorithm 6. Specifically, we add a new stage variable s. At each stage, we cut the current search direction into 2^s small blocks, and

Algorithm 4 Ray Searching Attack (Naive)

```
1: **input:** Model $f$, Original data example $\{\mathbf{x}, y\}$;
2: Initialize current best search direction $\mathbf{d}_{\text{best}} = (1, \ldots, 1)$
3: Initialize current best radius $r_{\text{best}} = \infty$
4: Initialize ray searching index $k = 1$
5: while remaining query budget $> 0$ do
6:     $\mathbf{d}_{\text{tmp}} = \mathbf{d}_{\text{best}}.\mathit{copy}()$
7:     $\mathbf{d}_{\text{tmp}}[k] = -\mathbf{d}_{\text{tmp}}[k]$
8:     $r_{\text{tmp}} = \text{DBR-Search}(f, \mathbf{x}, y, \mathbf{d}_{\text{tmp}}, r_{\text{best}})$
9:     if $r_{\text{tmp}} < r_{\text{best}}$ then
10:        $r_{\text{best}}, \mathbf{d}_{\text{best}} = r_{\text{tmp}}, \mathbf{d}_{\text{tmp}}$
11:    end if
12:    $k = k + 1$
13:    if $k == d$ then
14:        $k = 1$
15:    end if
16: end while
17: return $r_{\text{best}}, \mathbf{d}_{\text{best}}$
```

for each iteration, change the sign of the entire block simultaneously as the modified ray search direction for decision boundary radius search. After iterating through all blocks we move to the next stage and repeat the search process. Empirically speaking, Algorithm 6 largely improves the search efficiency by exploiting the spatial correlation mentioned above. All our experiments in Section 6.5 are conducted using Algorithm 6. Note that if the query budget is large enough, Algorithm 6 will, in the end, get to the case where the block size[3] equals to 1 and reduce to Algorithm 4 eventually.

[3]For completeness, when 2^s is larger than data dimension d, Algorithm 6 will only partition the search direction vector $\mathbf{d}_{\text{tmp}}$ into d blocks to ensure each block contain at least one dimension.

Algorithm 5 Decision Boundary Radius Search (DBR-Search)

```
1:  input: Model $f$, Original data example $\{\mathbf{x}, y\}$, Search direction $\mathbf{d}$, Current best radius $r_{\text{best}}$, Binary search tolerance $\epsilon$;
2:  Normalized search direction $\mathbf{d}_n = \mathbf{d}/\|\mathbf{d}\|_2$
3:  if $f(\mathbf{x} + r_{\text{best}} \cdot \mathbf{d}_n) == y$ then
4:      return $\infty$
5:  end if
6:  Set $start = 0, end = \min(r_{\text{best}}, \|\mathbf{d}\|_2)$
7:  while $end - start > \epsilon$ do
8:      $mid = (start + end)/2$
9:      if $f(x + r_{\text{best}} \cdot \mathbf{d}_n) == y$ then
9:          $end = mid$
10:     else
10:         $start = mid$
11:     end if
12: end while
13: return $end$
```

Note that all three algorithms (Algorithms 4, 5 and 6) do not involve any hyperparameters aside from the maximum number of queries, which is usually a predefined problem-related parameter. In sharp contrast, typical white-box attacks and zeroth-order optimization-based black-box attacks, need to tune quite a few hyperparameters in order to achieve good attack performance.

3.4 Experiments

In this section, we present the experimental results of our proposed Ray Searching attack (RayS). We first test RayS attack with other hard-label attack baselines on naturally trained models and then apply RayS attack on recently proposed state-of-the-art robust training models to test their performances. All of our experiments are conducted with NVIDIA 2080 Ti GPUs using Pytorch 1.3.1 on Python 3.6.9 platform.

3.4.1 Datasets and Target Models

We compare the performance of all attack algorithms on MNIST [LCB10], CIFAR-10 [KH$^+$09] and ImageNet [DDS$^+$09] datasets. Following adversarial examples literature [IEA$^+$18, MAS19, ADO20], we set $\epsilon = 0.3$ for MNIST dataset, $\epsilon = 0.031$ for CIFAR-10 dataset and $\epsilon = 0.05$ for ImageNet dataset. For naturally trained models, on the MNIST dataset, we attack two pre-trained 7-layer CNN: 4 convolutional layers followed by 3 fully connected layers with Max-pooling and RelU activation applied after each convolutional layer. The MNIST pre-trained model achieves 99.5% accuracy on the test set. On the CIFAR-10 dataset, we also use a 7-layer CNN structure with 4 convolutional layers and an additional 3 fully connected layers accompanied by Batchnorm and Max-pooling layers. The CIFAR-10 pre-trained model achieves 82.5% accuracy on the test set. For ImageNet experiments, we attack pre-trained ResNet-50 model [HZRS16b] and Inception V3 model [SVI$^+$16]. The pre-trained ResNet-50 model is reported to have a 76.2% top-1 accuracy. The pre-trained Inception V3 model is reported to have a 78.0% top-1 accuracy. For robust training models, we evaluate two well-recognized defenses: Adversarial Training (AdvTraining) [MMS$^+$18] and TRADES [ZYJ$^+$19]. In addition, we also test three other recently proposed defenses which claim to achieve the state-of-the-art robust accuracy: Sensible Adversarial Training (SENSE) [KW20], Feature Scattering-based Adversarial Training (FeatureScattering) [ZW19], Adversarial Interpolation Training (AdvInterpTraining) [ZX20]. Specifically, adversarial training [MMS$^+$18] solves a min-max optimization problem to minimize the adversarial loss. [ZYJ$^+$19] studied the trade-off between robustness and accuracy in adversarial training and proposed an empirically more robust model. [KW20] proposed to stop the attack generation when a valid attack has been found. [ZW19] proposed an unsupervised feature-scattering scheme for attack generation.

[ZX20] proposed an adversarial interpolation scheme for generating adversarial examples as well as adversarial labels and trained on those example-label pairs.

3.4.2 Baseline Methods

We compare the proposed algorithm with several state-of-the-art attack algorithms. Specifically, for attacking naturally trained models, we compare the proposed RayS attack with other hard-label attack baselines (i) OPT attack [CLC$^+$19], (ii) SignOPT attack [CSC$^+$20], and (iii) HSJA attack [CJW19]. We adopt the same hyperparameter settings in the original papers of OPT, SignOPT, and HSJA attack.

For attacking robust training models, we additionally compare with other state-of-the-art black-box attacks and even white-box attacks: (i) PGD attack [MMS$^+$18] (white-box), (ii) CW attack [CW17] [4] (white-box), (iii) SignHunter [ADO20] (black-box), and (iv) Square attack [ACFH19] (black-box). For PGD attack and CW attack, we set step size as 0.007 and provide attack results for 20 steps and also 100 steps. For SignHunter and Square attack, we adopt the same hyperparameter settings used in their original papers.

3.4.3 Comparison with hard-label Attack Baselines on Naturally Trained Models

In this subsection, we compare our Ray Searching attack with other hard-label attack baselines on naturally trained models. For each dataset (MNIST, CIFAR-10, and ImageNet), we randomly choose 1000 images from its test set that are verified to be correctly classified by the pre-trained model and test how many of them can be successfully attacked by the hard-label attacks. For each method, we restrict the maximum number of queries to 10000. For the sake of query efficiency, we stop the attack for a certain test sample once it is successfully attacked, i.e., the L_∞ norm distance between adversarial examples and original examples is

[4] To be precise, here CW attack refers to PGD updates with CW loss [CW17]

less than the pre-defined perturbation limit ϵ. Tables 3.1, 3.2, 3.3 and 3.4 present the performance comparison of all hard-label attacks on MNIST model, CIFAR-10 model, ResNet-50 Model and Inception V3 model respectively. For each experiment, we report the average and median of the number of queries needed for successful attacks for each attack, as well as the final attack success rate, i.e., the ratio of successful attacks against the total number of attack attempts. Specifically, on the MNIST dataset, we observe that our proposed RayS attack enjoys much better query efficiency in terms of average and median of the number of queries, and a much higher attack success rate than OPT and SignOPT methods. Note that the average (median) number of queries of SignOPT is larger than that of OPT. However, this does not mean that SignOPT performs worse than OPT. This result is due to the fact that the attack success rate of OPT is very low and its average (median) queries number is calculated based on the successfully attacked examples, which in this case, are the most vulnerable examples. HSJA attack, though improving over SignOPT[5], still falls behind our RayS attack. For the CIFAR model, the RayS attack still achieves the highest attack success rate. Though the HSJA attack comes close to the RayS attack in terms of attack success rate, its query efficiency still falls behind. On ResNet-50 and Inception V3 models, only RayS attack maintains the high attack success rate while the other baselines largely fall behind. Note that HSJA attack achieves similar or even slightly better average (median) queries on ImageNet models, suggesting that HSJA is efficient for the most vulnerable examples but not very effective when dealing with hard-to-attack examples. Figure 3.2 shows the attack success rate against the number of queries plot for all baseline methods on different models. Again we can see that the RayS attack overall achieves the highest attack success rate and best query efficiency compared with other hard-label attack baselines.

[5]Note that the relatively weak performance of SignOPT is due to the fact that SignOPT is designed for L_2 norm attack while this experiment is under the L_∞ norm setting. So the result does not conflict with the result reported in the original paper of SignOPT [CSC+20].

Table 3.1: Comparison of L_∞ norm based hard-label attack on MNIST dataset ($\epsilon = 0.3$).

Methods	Avg. Queries	Med. Queries	ASR (%)
OPT	3260.9	2617.0	20.9
SignOPT	3784.3	3187.5	62.8
HSJA	161.6	154.0	91.2
RayS	**107.0**	**47.0**	**100.0**
PGD (white-box)	-	-	100.0

Table 3.2: Comparison of L_∞ norm based hard-label attack on CIFAR-10 dataset ($\epsilon = 0.031$).

Methods	Avg. Queries	Med. Queries	ASR (%)
OPT	2253.3	1531.0	31.0
SignOPT	2601.3	1649.0	60.1
HSJA	1021.6	714.0	99.7
RayS	**792.8**	**343.5**	**99.8**
PGD (white-box)	-	-	100.0

3.4.4 Evaluating the Robustness of State-of-the-art Robust Models

In this subsection, we further test our proposed Ray Searching attack by applying it to the state-of-the-art robust training models. Specifically, we selected five recently proposed open-sourced defenses on the CIFAR-10 dataset and WideResNet [ZK16] architecture. For the test examples, we randomly choose 1000 images from the CIFAR-10 test set. We set the maximum number of queries as 40000.

In terms of evaluation metrics, following the literature of robust training [MMS$^+$18, ZYJ$^+$19], we report the natural accuracy and robust accuracy (classification accuracy under adversarial attacks) of the defense model. In addition, we report a new metric called *Average Decision Boundary Distance* (ADBD), which is defined as the average L_∞ norm distance

Table 3.3: Comparison of L_∞ norm based hard-label attack on ImageNet dataset for ResNet-50 model ($\epsilon = 0.05$).

Methods	Avg. Queries	Med. Queries	ASR (%)
OPT	1344.5	655.5	14.2
SignOPT	3103.5	2434.0	36.0
HSJA	749.6	**183.0**	19.9
RayS	**574.0**	296.0	**99.8**
PGD (white-box)	-	-	100.0

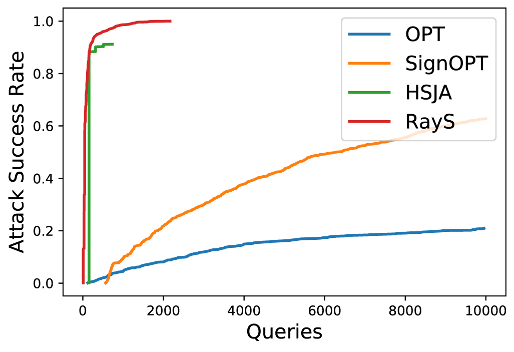
Table 3.4: Comparison of L_∞ norm based hard-label attack on ImageNet dataset for Inception V3 model ($\epsilon = 0.05$).

Methods	Avg. Queries	Med. Queries	ASR (%)
OPT	2375.6	1674.0	21.9
SignOPT	2624.8	1625.0	39.9
HSJA	**652.3**	**362.0**	23.7
RayS	748.2	370.0	**98.9**
PGD (white-box)	-	-	100.0

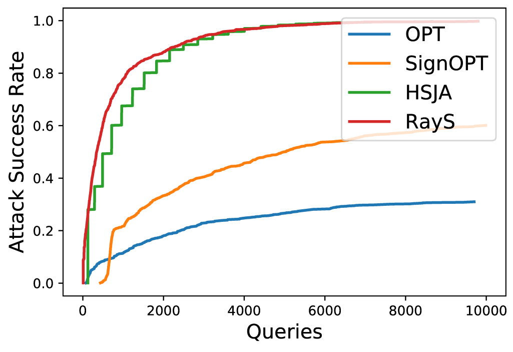
between *all* test examples to their nearest decision boundaries. Note that ADBD is not valid for white-box and black-box attacks that follow formulation (3.3.1), since they cannot find the nearest decision boundaries for all test examples.

Here we want to emphasize the difference between ADBD and the average L_∞ distortion in the adversarial learning literature. Note that L_∞ distortion[6] usually refers to the L_∞ norm distance between successful adversarial attack examples and their corresponding original clean examples and therefore, is affected by the choice of the maximum perturbation limit ϵ.

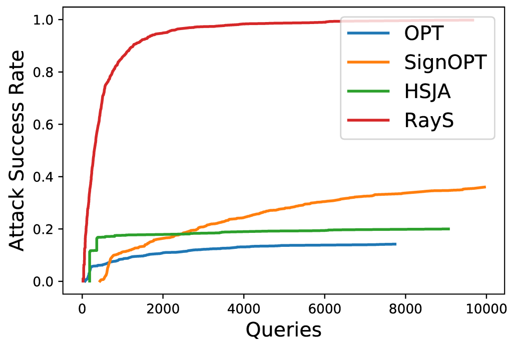
[6]For all white-box and black-box attacks tested in this experiment, their L_∞ distortions are very close to 0.031, which is the perturbation limit ϵ. Therefore, we do not report the L_∞ distortion in the tables as it does not provide much additional information.



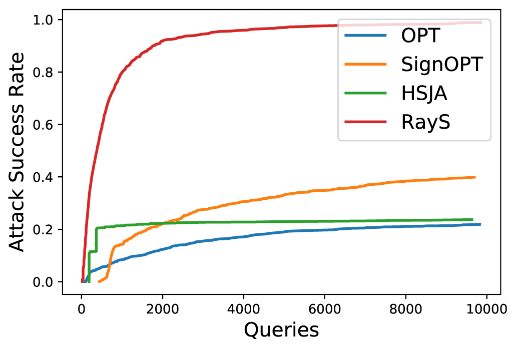
(a) MNIST



(b) CIFAR



(c) ResNet-50



(d) Inception V3

Figure 3.2: Attack success rate against the number of queries plots for different hard-label attacks on MNIST, CIFAR-10 and ImageNet datasets.

For hard-label attacks, only considering the attacks with a radius less than ϵ loses too much information and cannot capture the whole picture of model robustness[7]. On the other hand, the ADBD metric, though only valid for hard-label attacks, provides a meaningful estimation on the average distance from the original clean examples to their decision boundaries.

Tables 3.5, 3.6, 3.7, 3.8 and 3.9 show the comparison of different adversarial attack methods on five selected robust models. Specifically, for two well recognized robust training models, Adversarial Training (in Table 3.5) and TRADES (in Table 3.6), we observe that white-box attacks are still the strongest attacks, where PGD attack and CW attack achieve very similar attack performances. For black-box attacks, the SignHunter attack and Square

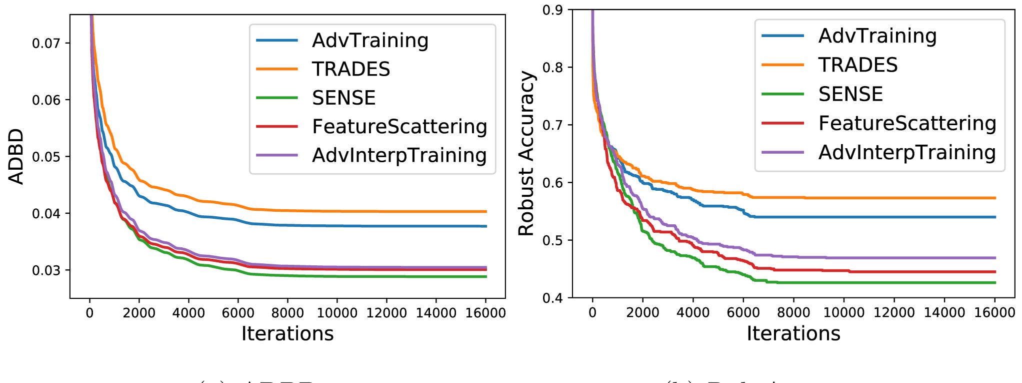
[7]For hard-label attacks, the ADBD value is always larger than the L_∞ distortion.

attack achieve similar attack performances as their white-box counterparts. In terms of hard-label attacks, our proposed RayS attack also achieves comparable attack performance as black-box or even white-box attacks given the most restricted access to the target model. When comparing with other hard-label attack baselines, it can be seen that our RayS attack achieves significant performance improvement in terms of both robust accuracy (over 20%) and the average decision boundary distance (reduced by 30%). The less effectiveness in attacking L_∞ norm threat model makes the SignOPT attack and HSJA attack less practical. For the Sensible Adversarial Training model (in Table 3.7), it indeed achieves overall better robust accuracy under white-box attacks, compared with Adversarial Training and TRADES. For black-box attacks, the SignHunter attack achieves similar performance as the PGD attack and the Square attack achieves similar performance as CW attacks. Interestingly, we observe that for hard-label attacks, our proposed RayS attack achieves 42.5% robust accuracy, reducing 20% from PGD attack and 15% from CW attack, suggesting that the robustness of Sensible Adversarial Training is not truly better than TRADES and Adversarial Training, but just looks better under PGD attack and CW attack. For Feature Scattering-based Adversarial Training model (in Table 3.8), note that the CW attack is much more effective than the PGD attack. Also for black-box attacks, the performance of the Square attack is much better than SignHunter attack[8], suggesting that the CW loss is more effective than CrossEntropy loss in attacking Feature Scattering-based Adversarial Training model. Again, we can observe that our proposed RayS attack reduces the robust accuracy of the PGD attack by 28% and CW attack by 10%. This also suggests that Feature Scattering-based Adversarial Training model does not really provide better robustness than Adversarial Training or TRADES. For the Adversarial Interpolation Training model (in Table 3.9), under white-box attacks, it achieves surprisingly high robust accuracy of 75.3% (under the PGD attack) and 68.9% (under the CW attack), and similar results can be obtained under the corresponding black-box attacks. However, it is still not truly robust under our RayS attack,

[8]Square attack is based on CW loss while the SignHunter attack is based on CrossEntropy loss.

reducing the robust accuracy of the PGD attack by 28% and the CW attack by 22%. Note that in this experiment, the HSJA attack also achieves lower robust accuracy than the PGD attack, suggesting that all hard-label attacks may have the potential to detect those falsely robust models that deceive current white-box/black-box attacks, but the low efficiency of HSJA restricts its power for greater use.

To obtain the overall comparison on the robustness of the five selected robust training models under our proposed RayS attack, we plot the Average Decision Boundary Distance (ADBD) against RayS attack iterations and the robust accuracy against RayS attack iterations in Figure 3.3. First, it can be seen that the Average Decision Boundary Distance and robust accuracy indeed converge and remain stable after around 10000 RayS attack iterations. Figure 3.3 suggests that among the five selected robust training models, TRADES and Adversarial Training remain the most robust models while Sensible Adversarial Training, Feature Scattering-based Adversarial Training and Adversarial Interpolation Training, are not as robust as they appear under PGD attacked and CW attack. Note also that even though Sensible Adversarial Training, Feature Scattering-based Adversarial Training and Adversarial Interpolation Training have quite different robust accuracy results under RayS attack, their ADBD results are quite similar.



(a) ADBD (b) Rob Accuracy

Figure 3.3: Average Decision Boundary Distance (ADBD) and Robust accuracy against RayS attack iterations plot for several robust models.

Table 3.5: Comparison of different adversarial attack methods on Adversarial Training [MMS+18] for CIFAR-10 dataset (WideResNet, $\epsilon = 0.031$, natural accuracy: 87.4%).

Methods	Att. Type	ADBD	Rob. Acc (%)
SignOPT	hard-label	0.202	85.1
HSJA	hard-label	0.060	76.8
RayS	hard-label	**0.038**	**54.0**
SignHunter	black-box	-	**50.9**
Square	black-box	-	52.7
PGD-20	white-box	-	51.1
CW-20	white-box	-	51.8
PGD-100	white-box	-	**50.6**
CW-100	white-box	-	51.5

3.5 Discussions and Conclusions

In this paper, we proposed the Ray Searching attack, which only requires the hard-label output of the target model. The proposed Ray Searching attack is much more effective in attack success rate and efficient in terms of query complexity, compared with other hard-label attacks. Moreover, it can be used as a sanity check tool for possible "falsely robust" models that deceive current white-box and black-box attacks.

In the following discussions, we try to analyze the key ingredients for the success of the proposed Ray Searching attack.

Why RayS attack is more effective and efficient than the other hard-label baselines?

As we mentioned before, traditional hard-label attacks are more focused on the L_2 norm threat model with only a few extensions to the L_∞ norm threat model. While for our RayS attack, we reformulate the continuous problem of finding the closest decision boundary into a discrete problem based on empirical findings in L_∞ norm threat model, which leads to a

Table 3.6: Comparison of different adversarial attack methods on TRADES [ZYJ+19] for CIFAR-10 dataset (WideResNet, $\epsilon = 0.031$, natural accuracy: 85.4%).

Methods	Att. Type	ADBD	Rob. Acc (%)
SignOPT	hard-label	0.196	84.0
HSJA	hard-label	0.064	71.6
RayS	hard-label	**0.040**	**57.3**
SignHunter	black-box	-	**56.1**
Square	black-box	-	**56.1**
PGD-20	white-box	-	56.5
CW-20	white-box	-	55.6
PGD-100	white-box	-	56.3
CW-100	white-box	-	**55.3**

more effective hard-label attack. On the other hand, the strategy of directly searching for the closest decision boundary together with a fast check step eliminates unnecessary searches and significantly improves the attack efficiency.

Why RayS attack can detect possible "false" robust models while traditional white-box and black-box attacks cannot?

One thing we observe from Section 6.5 is that although different attacks lead to different robust accuracy results, their attack performances are correlated with the choice of attack loss functions, e.g., both PGD attack and SignHunter attack utilize CrossEntropy loss and their attack performances are similar in most cases. A similar effect can also be seen for the CW attack and Square attack, both of which utilize the CW loss function. However, these loss functions were used as surrogate losses to problem (3.3.1), and they may not be able to truly reflect the quality/potential of an intermediate example (an example near the original clean example that is not yet a valid adversarial example). For instance, consider the case where two intermediate examples share the same log probability at ground truth

Table 3.7: Comparison of different adversarial attack methods on SENSE [KW20] for CIFAR-10 dataset (WideResNet, $\epsilon = 0.031$, natural accuracy: 91.9%).

Methods	Att. Type	ADBD	Rob. Acc (%)
SignOPT	hard-label	0.170	88.2
HSJA	hard-label	0.044	66.6
RayS	hard-label	**0.029**	**42.5**
SignHunter	black-box	-	61.9
Square	black-box	-	**58.2**
PGD-20	white-box	-	62.1
CW-20	white-box	-	59.7
PGD-100	white-box	-	60.1
CW-100	white-box	-	**57.9**

class y, but vary drastically on other classes. Their CrossEntropy losses are the same in such cases, but one may have a larger potential to develop into a valid adversarial example than the other one (e.g., the second-largest probability is close to the largest probability). Therefore, CrossEntropy loss does not really reflect the true quality/potential of the intermediate examples. Similar instances can also be constructed for CW loss. In sharp contrast, our RayS attack considers the decision boundary radius as the search criterion[9]. When we compare two examples on the decision boundary, it is clear that the closer one is better. In cases where the attack problem is hard to solve and the attacker could easily get stuck at intermediate examples (e.g., attacking robust training models), it is easy to see that the RayS attack stands a better chance of finding a successful attack. This partially explains the superiority of the RayS attack in detecting "falsely robust" models.

[9]Actually it is a criterion for all hard-label attack.

Table 3.8: Comparison of different adversarial attack methods on Feature-Scattering [ZW19] for CIFAR-10 dataset (WideResNet, $\epsilon = 0.031$, natural accuracy: 91.3%).

Methods	Att. Type	ADBD	Rob. Acc (%)
SignOPT	hard-label	0.175	87.1
HSJA	hard-label	0.048	70.0
RayS	hard-label	**0.030**	**44.5**
SignHunter	black-box	-	67.3
Square	black-box	-	**55.3**
PGD-20	white-box	-	72.8
CW-20	white-box	-	57.2
PGD-100	white-box	-	70.4
CW-100	white-box	-	**54.8**

Table 3.9: Comparison of different adversarial attack methods on Adversarial Interpolation Training [ZX20] for CIFAR-10 dataset (WideResNet, $\epsilon = 0.031$, natural accuracy: 91.0%).

Methods	Att. Type	ADBD	Rob. Acc (%)
SignOPT	hard-label	0.169	84.2
HSJA	hard-label	0.049	70.5
RayS	hard-label	**0.031**	**46.9**
SignHunter	black-box	-	73.6
Square	black-box	-	**69.0**
PGD-20	white-box	-	75.6
CW-20	white-box	-	69.2
PGD-100	white-box	-	75.3
CW-100	white-box	-	**68.9**

Algorithm 6 Ray Searching Attack (Hierarchical)

```
1: **input:** Model $f$, Original data example $\{\mathbf{x}, y\}$;
2: Initialize current best search direction $\mathbf{d}_{\text{best}} = (1, \ldots, 1)$
3: Initialize current best radius $r_{\text{best}} = \infty$
4: Initialize stage $s = 0$
5: Initialize block index $k = 1$
6: **while** remaining query budget $> 0$ **do**
7:     $\mathbf{d}_{\text{tmp}} = \mathbf{d}_{\text{best}}.copy()$
8:     Cut $\mathbf{d}_{\text{tmp}}$ into $2^s$ blocks and denote index set in the $k$-th block by $\mathcal{I}_k$
9:     $\mathbf{d}_{\text{tmp}}[\mathcal{I}_k] = -\mathbf{d}_{\text{tmp}}[\mathcal{I}_k]$
10:    $r_{\text{tmp}} = \text{DBR-Search}(f, \mathbf{x}, y, \mathbf{d}_{\text{tmp}}, r_{\text{best}})$
11:    **if** $r_{\text{tmp}} < r_{\text{best}}$ **then**
12:        $r_{\text{best}}, \mathbf{d}_{\text{best}} = r_{\text{tmp}}, \mathbf{d}_{\text{tmp}}$
13:    **end if**
14:    $k = k + 1$
15:    **if** $k == 2^s$ **then**
16:        $s = s + 1$
17:        $k = 1$
18:    **end if**
19: **end while**
20: return $r_{\text{best}}, \mathbf{d}_{\text{best}}$
```

CHAPTER 4

Understanding the Intrinsic Robustness of Image Distributions using Conditional Generative Models

4.1 Introduction

Deep neural networks (DNNs) have achieved remarkable performance on many visual [SHK12, HZRS16a] and speech [HDY$^+$12] recognition tasks, but recent studies have shown that state-of-the-art DNNs are surprisingly vulnerable to *adversarial perturbations*, small imperceptible input transformations that are designed to switch the prediction of the classifier [SZS$^+$14, GSS15]. This has led to a vigorous arms race between heuristic defenses [PMW$^+$16, MMS$^+$18, CAD$^+$18, WMB$^+$19] that propose ways to defend against existing attacks and newly-devised attacks [CW17, ACW18, TCBM20] that are able to penetrate such defenses. Reliable defenses appear to be elusive, despite progress on provable defenses, including formal verification [KBD$^+$17, TXT19] and relaxation-based certification methods [SND18, RSL18, WK18, GDS$^+$19, WCAJ18]. Even the strongest of these defenses leave large opportunities for adversaries to find adversarial examples, while suffering from high computation costs and scalability issues.

Witnessing the difficulties of constructing robust classifiers, a line of recent works [GMF$^+$18, FFF18a, MDM19, SHS$^+$19] aims to understand the limitations of robust learning by providing theoretical bounds on adversarial robustness for arbitrary classifiers. By imposing different assumptions on the underlying data distributions and allowable perturbations, all of these theoretical works show that no adversarially robust classifiers exist for an assumed metric probability space, as long as the perturbation strength is sublinear in the typical norm of the inputs. Although such impossibility results seem disheartening to the goal of building robust classifiers, it remains unknown to what extent real image distributions satisfy the assumptions needed to obtain these results.

In this paper, we aim to bridge the gap between the theoretical robustness analyses on well-behaved data distributions and the maximum achievable adversarial robustness, which we call *intrinsic robustness* (formally defined by Definition 4.3.2), for typical image distributions. More specifically, we assume the underlying data lie on a separable low-dimensional manifold, which can be captured using a conditional generative model, then systematically study the intrinsic robustness based on the conditional generating process from both theoretical and experimental perspectives. Our main contributions are:

- We prove a fundamental bound on intrinsic robustness (Section 4.4), provided that the underlying data distribution can be captured by a conditional generative model, solving an open problem in [FFF18a].
- Building upon a trained conditional generative model that mimics the underlying data generating process, we empirically evaluate the intrinsic robustness on image distributions based on MNIST and ImageNet (Section 4.7.2). Our estimates of intrinsic robustness demonstrate that there is still a large gap between the limits implied by our theory and the state-of-the-art robustness achieved by robust training methods (Section 4.7.3).
- We theoretically characterize the fundamental relationship between the *in-distribution adversarial risk* (which restricts adversarial examples to lie on the image manifold, and is formally defined by Definition 4.3.3) and the intrinsic robustness (Remark 4.4.6), and propose an optimization method to search for in-distribution adversarial examples with respect to a given classifier. Our estimated in-distribution robustness for state-of-the-art adversarially trained classifiers, together with the derived intrinsic robustness

bound, provide a better understanding on the intrinsic robustness for natural image distributions (Section 4.7.4).

Notation: We use lower boldfaced letters such as $\boldsymbol{x}$ to denote vectors, and $[n]$ to denote the index set $\{1, 2, \ldots, n\}$. For any $\boldsymbol{x} \in \mathcal{X}$ and $\epsilon \geq 0$, denote by $\mathcal{B}(\boldsymbol{x}, \epsilon, \Delta) = \{\boldsymbol{x}' \in \mathcal{X} : \Delta(\boldsymbol{x}, \boldsymbol{x}') \leq \epsilon\}$ the ϵ-ball around $\boldsymbol{x}$ with radius ϵ in some distance metric Δ. When the metric is free of context, we simply write $\mathcal{B}(\boldsymbol{x}, \epsilon) = \mathcal{B}(\boldsymbol{x}, \epsilon, \Delta)$. We use $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ to denote the d-dimensional standard Gaussian distribution, and let ν_d be its probability measure. For the one dimensional case, we use $\Phi(x)$ to denote the cumulative distribution function (CDF) of $\mathcal{N}(0, 1)$, and use $\Phi^{-1}(x)$ to denote its inverse function. For any function $g : \mathcal{Z} \rightarrow \mathcal{X}$ and probability measure ν defined over $\mathcal{Z}$, $g_*(\nu)$ denotes the push-forward measure of ν. The ℓ_2-norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is defined as $\|\boldsymbol{x}\|_2 = (\sum_{i\in[n]} x_i^2)^{1/2}$.

4.2 Related Work

Several recent works [GMF$^+$18, MDM19, SHS$^+$19, Doh19, BCM19] derived theoretical bounds on maximum achievable adversarial robustness using isoperimetric inequality under different assumptions of the input space. For instance, based on the assumption that the input data are uniformly distributed over two concentric n-spheres [GMF$^+$18] or the underlying metric probability space satisfies a concentrated property [MDM19], any classifier with constant test error was proven to be vulnerable to adversarial perturbations sublinear to the input dimension. [SHS$^+$19] showed that adversarial examples are inevitable, provided the maximum density of the underlying input distribution is small relative to uniform density. However, none of the above theoretical works provide any experiments to justify the imposed assumptions hold for real datasets, thus it is unclear whether the derived theoretical bounds are meaningful for typical image distributions. Our work belongs to this line of research, but encompasses the practical goal of understanding the robustness limits for real image distributions.

The most related literature to ours is [FFF18a], which proved a classifier-independent upper bound on intrinsic robustness, provided the underlying distribution is well captured by a smoothed generative model with Gaussian latent space and small Lipschitz parameter. However, their proposed theory cannot be applied to image distributions that lie on a low-dimensional, non-smooth manifold, as their framework requires examples from different classes to be close enough in the latent space. In contrast, our proposed theoretical bounds on intrinsic robustness are more general in that they can be applied to non-smoothed data manifolds, such as image distributions generated by conditional models. In addition, we propose an empirical method to estimate the intrinsic robustness on the generated image distributions under worst-case ℓ_2 perturbations.

[MZME19] proposed to understand the inherent limitations of robust learning using heuristic methods to measure the concentration of measure based on a given set of i.i.d. samples. However, it is unclear to what extent the estimated sample-based concentration approximates the actual intrinsic robustness with respect to the underlying data distribution. In comparison, we assume the underlying data distribution can be captured by a conditional generative model and directly study the robustness limit on the generated data distribution.

4.3 Preliminaries

We focus on the task of image classification. Let $(\mathcal{X}, \mu, \Delta)$ be a metric probability space, where $\mathcal{X} \subseteq \mathbb{R}^n$ denotes the input space, μ is a probability distribution over $\mathcal{X}$ and Δ is some distance metric defined on $\mathcal{X}$. Suppose there exists a ground-truth function, $f^* : \mathcal{X} \to [K]$, that gives a label to any image $\boldsymbol{x} \in \mathcal{X}$, where $[K]$ denotes the set of all possible class labels. The objective of classification is to learn a function $f : \mathcal{X} \to [K]$ that approximates f^* well. In the context of adversarial examples, f is typically evaluated based on *risk*, which captures the classification accuracy of f on normal examples, and *adversarial risk*, which captures the classifier's robustness against adversarial perturbations:

Definition 4.3.1. Let $(\mathcal{X}, \mu, \Delta)$ be a metric probability space and f^* be the ground-truth classifier. For any classifier f, the *risk* of f is defined as:

$$\mathrm{Risk}_\mu(f) = \Pr_{\boldsymbol{x}\sim\mu}\big[f(\boldsymbol{x}) \neq f^*(\boldsymbol{x})\big].$$

The *adversarial risk* of f against perturbations with strength ϵ in metric Δ is defined as:

$$\mathrm{AdvRisk}_\mu^\epsilon(f) = \Pr_{\boldsymbol{x}\sim\mu}\big[\exists\, \boldsymbol{x}' \in \mathcal{B}(\boldsymbol{x}, \epsilon) \text{ s.t.} f(\boldsymbol{x}') \neq f^*(\boldsymbol{x}')\big].$$

Other definitions of adversarial risk also exist in literature, such as the definition used in [MMS+18] and the one proposed in [FFF18a]. However, these definitions are equivalent to each other under the assumption that small perturbations do not change the ground-truth labels. Another closely-related definition for adversarial robustness is the expected distance to the nearest error (see [DMM18] for the relation between these definitions). Our results can be applied to this definition as well.

Under different assumptions of the metric probability space, previous works proved model-independent bounds on adversarial robustness. *Intrinsic robustness*, defined originally by [MZME19], captures the maximum adversarial robustness that can be achieved for a given robust learning problem:

Definition 4.3.2. Using the same settings as in Definition 4.3.1 and let $\mathcal{F}$ be some class of classifiers. The *intrinsic robustness* with respect to $\mathcal{F}$ is defined as:

$$\mathrm{Rob}_\mu^\epsilon(\mathcal{F}) = 1 - \inf_{f\in\mathcal{F}}\big\{\mathrm{AdvRisk}_\mu^\epsilon(f)\big\}.$$

In this work, we consider the class of imperfect classifiers that have risk at least some $\alpha > 0$.

Motivated by the great success of producing natural-looking images using conditional generative adversarial nets (GANs) [MO14, OOS17, BDS19], we assume the underlying data distribution μ can be modeled by some *conditional generative model*. A generative model can be seen as a function $g : \mathcal{Z} \to \mathcal{X}$ that maps some latent distribution, usually assumed to be multivariate Gaussian, to some generated distribution over $\mathcal{X}$.

Conditional generative models incorporate the additional class information into the data generating process. A conditional generative model can be considered as a set of generative models $\{g_i\}_{i\in[K]}$, where images from the i-th class can be generated by transforming latent Gaussian vectors through g_i. More rigorously, we say a probability distribution μ can be generated by a conditional generative model $\{(g_i, p_i)\}_{i\in[K]}$, if $\mu = \sum_{i=1}^{K} p_i \cdot (g_i)_*(\nu_d)$, where K is the total number of different class labels, and $p_i \in [0, 1]$ represents the probability of sampling an image from class i.

Based on the conditional generative model, we introduce the definition of *in-distribution adversarial risk*:

Definition 4.3.3. Consider the same settings as in Definition 4.3.1. Suppose μ can be captured by a conditional generative model $\{(g_i, p_i)\}_{i\in[K]}$. For any given classifier f, the *in-distribution adversarial risk* of f against ϵ-perturbations is defined as:

$$\text{In-AdvRisk}_\mu^\epsilon(f) = \Pr_{(\boldsymbol{x},i)\sim\mu}\big[\exists\ \boldsymbol{z}' \in \mathcal{Z} \text{ s.t. } g_i(\boldsymbol{z}') \in \mathcal{B}(\boldsymbol{x}, \epsilon) \text{ and } f(g_i(\boldsymbol{z}')) \neq f^*(g_i(\boldsymbol{z}'))\big].$$

Given the fact that the in-distribution adversarial risk restricts the adversarial examples to be on the image manifold, it holds that, for any classifier f, $\text{In-AdvRisk}_\mu^\epsilon(f) \leq \text{AdvRisk}_\mu^\epsilon(f)$. As will be shown in the next section, such a notion of in-distribution adversarial risk is closely related to the intrinsic robustness for the considered class of imperfect classifiers.

4.4 Main Theoretical Results

In this section, we present our main theoretical results on intrinsic robustness, provided the underlying distribution can be modeled by some conditional generative model (our results and proof techniques could also be easily applied to unconditional generative models). Based on the underlying generative process, the following local Lipschitz condition connects perturbations in the image space to the latent space.

Condition 4.4.1. Let $g : \mathbb{R}^d \to \mathcal{X}$ be a generative model that maps the latent Gaussian distribution ν_d to some generated distribution. Consider Euclidean distance as the distance metric for $\mathbb{R}^d$, and Δ as the metric for $\mathcal{X}$. Given $r > 0$, g is said to be $L(r)$-locally Lipschitz with probability at least $1 - \delta$, if it satisfies

$$\Pr_{\boldsymbol{z} \sim \nu_d} \Big[\forall \boldsymbol{z}' \in \mathcal{B}(\boldsymbol{z}, r), \Delta\big(g(\boldsymbol{z}'), g(\boldsymbol{z})\big) \leq L(r) \|\boldsymbol{z}' - \boldsymbol{z}\|_2 \Big] \geq 1 - \delta.$$

As the main tool for bounding the intrinsic robustness, we present the Gaussian Isoperimetric inequality for the sake of completeness. This inequality, proved by [Bor75] and [ST78], bounds the minimum expansion of any subset with respect to the standard Gaussian measure.

Lemma 4.4.2 (Gaussian Isoperimetric Inequality)**.** Consider metric probability space $(\mathbb{R}^d, \nu_d, \|\cdot\|_2)$, where ν_d is the probability measure for d-dimensional standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, and $\|\cdot\|_2$ denotes the Euclidean distance. For any subset $\mathcal{E} \subseteq \mathbb{R}^d$ and $r \geq 0$, let $\mathcal{E}_r = \big\{\boldsymbol{z} \in \mathbb{R}^d : \exists \boldsymbol{z}' \in \mathcal{E}, \text{ s.t. } \|\boldsymbol{z} - \boldsymbol{z}'\|_2 \leq r\big\}$ be the r-expansion of $\mathcal{E}$, then it holds that

$$\nu_d(\mathcal{E}_r) \geq \Phi\big(\Phi^{-1}\big(\nu_d(\mathcal{E})\big) + r\big), \tag{4.4.1}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-u^2/2) \cdot du$ is the CDF of $\mathcal{N}(0,1)$, and $\Phi^{-1}(x)$ denotes its inverse function.

In particular, when $\mathcal{E}$ belongs to the set of half-spaces, the equality is achieved in (4.4.1).

Making use of the Gaussian Isoperimetric Inequality and the local Lipschitz condition of the conditional generator, the following theorem proves a lower bound on the (in-distribution) adversarial risk for any given classifier, provided the underlying distribution can be captured by a conditional generative model.

Theorem 4.4.3. Let $(\mathcal{X}, \mu, \Delta)$ be a metric probability space and $f^* : \mathcal{X} \to [K]$ be the underlying ground-truth. Suppose μ can be generated by a conditional generative model $\{(g_i, p_i)\}_{i \in [K]}$. Given $\epsilon > 0$, suppose there exist constants $r > 0$ and $\delta \in (0, 1]$ such that for any $i \in [K]$, g_i satisfies $L_i(r)$-local Lipschitz property with probability at least $1-\delta$ and $r \cdot L_i(r) \geq \epsilon$. Then for any classifier f, it holds that

$$\mathrm{AdvRisk}_\mu^\epsilon(f) \geq \text{In-AdvRisk}_\mu^\epsilon(f) \geq \sum_{i=1}^K p_i \cdot \Phi\bigg(\Phi^{-1}\big(\mathrm{Risk}_{\mu_i}(f)\big) + \frac{\epsilon}{L_i(r)}\bigg) - \delta,$$

where $\mu_i = (g_i)_*(\nu_d)$ is the pushforward measure of ν_d though g_i, for any $i \in [K]$.

We provide a proof in Appendix 4.5.1. Theorem 4.4.3 suggests the (in-distribution) adversarial risk is related to the risk on each data manifold and the ratio between the perturbation strength and the Lipschitz constant.

The following theorem, proved in Appendix 4.5.2, gives a theoretical upper bound on the intrinsic robustness with respect to the class of imperfect classifiers.

Theorem 4.4.4. Under the same setting as in Theorem 4.4.3, let $L_{\max}(r) = \max_{i\in[K]} L_i(r)$. Consider the class of imperfect classifiers $\mathcal{F}_\alpha = \{f : \mathrm{Risk}_\mu(f) \geq \alpha\}$ with $\alpha > 0$, then the intrinsic robustness with respect to $\mathcal{F}_\alpha$ can be bounded as,

$$\mathrm{Rob}_\mu^\epsilon(\mathcal{F}_\alpha) \leq 1 + \delta - \min_{i\in[K]} \bigg\{p_i \cdot \Phi\bigg(\Phi^{-1}\Big(\frac{\alpha}{p_i}\Big) + \frac{\epsilon}{L_{\max}(r)}\bigg)\bigg\},$$

provided that $\alpha/p_i \leq 1$ for any $i \in [K]$. In addition, if we consider the family of classifiers that have conditional risk at least α for each class, namely $\widetilde{\mathcal{F}}_\alpha = \{f : \mathrm{Risk}_{\mu_i}(f) \geq \alpha, \forall i \in [K]\}$, then the intrinsic robustness with respect to $\widetilde{\mathcal{F}}_\alpha$ can be bounded by

$$\mathrm{Rob}_\mu^\epsilon(\widetilde{\mathcal{F}}_\alpha) \leq 1 + \delta - \sum_{i=1}^K p_i \cdot \Phi\bigg(\Phi^{-1}(\alpha) + \frac{\epsilon}{L_{\max}(r)}\bigg).$$

Remark 4.4.5. Theorem 4.4.4 shows that if the data distribution can be captured by a conditional generative model, the intrinsic robustness bound with respect to imperfect classifiers will largely depend on the ratio $\epsilon/L_{\max}$. For instance, if we assume the ratio $\epsilon/L_{\max} = 1$, then Theorem 4.4.4 suggests that no classifier with initial risk at least 5% can achieve robust accuracy exceeding 75% for the assumed data generating process. In addition, if we assume the local Lipschitz parameter $L_{\max}$ is some constant, then adversarial robustness is indeed not achievable for high-dimensional data distributions, provided the perturbation strength ϵ is sublinear to the input dimension, which is the typical setting considered.

Remark 4.4.6. The intrinsic robustness is closely related to the in-distribution adversarial risk. For the class of classifiers $\mathcal{F}_\alpha$, one can prove that the intrinsic robustness is equivalent to the maximum achievable in-distribution adversarial robustness:

$$\mathrm{Rob}_\mu^\epsilon(\mathcal{F}_\alpha) = 1 - \inf_{f\in\mathcal{F}_\alpha}\{\text{In-AdvRisk}_\mu^\epsilon(f)\}. \tag{4.4.2}$$

Trivially, $\mathrm{AdvRisk}_\mu^\epsilon(f) \geq \text{In-AdvRisk}_\mu^\epsilon(f)$ holds for any f. For a given $f \in \mathcal{F}_\alpha$, one can construct an $h_f \in \mathcal{F}_\alpha$ such that $h_f(\boldsymbol{x}) = f(\boldsymbol{x})$ if $\boldsymbol{x} \in \mathcal{E}_f \cap \mathcal{M}$ and $h_f(\boldsymbol{x}) = f^*(\boldsymbol{x})$ otherwise, where $\mathcal{E}_f = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) \neq f^*(\boldsymbol{x})\}$ denotes the error region of f and $\mathcal{M}$ is the considered image manifold. The construction immediately suggests $\text{In-AdvRisk}_\mu^\epsilon(f) = \mathrm{AdvRisk}_\mu^\epsilon(h_f)$, which implies,

$$\inf_{f\in\mathcal{F}_\alpha}\{\text{In-AdvRisk}_\mu^\epsilon(f)\} = \inf_{f\in\mathcal{F}_\alpha}\{\mathrm{AdvRisk}_\mu^\epsilon(h_f)\} \geq \inf_{f\in\mathcal{F}_\alpha}\{\mathrm{AdvRisk}_\mu^\epsilon(f)\}.$$

Combining both directions proves the soundness of (4.4.2). This equivalence suggests the in-distribution adversarial robustness of any classifier in $\mathcal{F}_\alpha$ can be viewed as a lower bound on the actual intrinsic robustness, which motivates us to study the intrinsic robustness by estimating the in-distribution adversarial robustness of trained robust models in our experiments.

4.5 Proof of Main Theorem

This section presents the detailed proofs of Theorems 4.4.3 and 4.4.4 in Section 4.4.

4.5.1 Proof of Theorem 4.4.3

Proof. Let $\mathcal{E} = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) \neq f^*(\boldsymbol{x})\}$ be the error region in the image space and $\mathcal{E}_\epsilon = \{\boldsymbol{x} \in \mathcal{X} : \Delta(\boldsymbol{x}, \mathcal{E}) \leq \epsilon\}$ be the ϵ-expansion of $\mathcal{E}$ in metric Δ. By Definition 4.3.1, we have

$$\mathrm{AdvRisk}_\mu^\epsilon(f) = \mu(\mathcal{E}_\epsilon) = \sum_{i=1}^K p_i \cdot \mu_i(\mathcal{E}_\epsilon) = \sum_{i=1}^K p_i \cdot \mathrm{AdvRisk}_{\mu_i}^\epsilon(f).$$

Since according to Definition 4.3.3, we have $\mathrm{AdvRisk}_{\mu_i}^\epsilon(f) \geq \text{In-AdvRisk}_{\mu_i}^\epsilon(f)$ for any $i \in [K]$. Thus, it remains to lower bound each term $\text{In-AdvRisk}_{\mu_i}^\epsilon(f)$ individually. For any classifier f, we have

$$\begin{aligned}\text{In-AdvRisk}_{\mu_i}^\epsilon(f) &= \Pr_{\boldsymbol{z}\sim\nu_d}\Big[\exists\, \boldsymbol{z}' \in \mathbb{R}^d,\ \text{s.t. } \Delta\big(g_i(\boldsymbol{z}'), g_i(\boldsymbol{z})\big) \leq \epsilon \text{ and } f\big(g_i(\boldsymbol{z}')\big) \neq f^*\big(g_i(\boldsymbol{z}')\big)\Big] \\ &\geq \underbrace{\Pr_{\boldsymbol{z}\sim\nu_d}\Big[\exists\, \boldsymbol{z}' \in \mathcal{B}\big(\boldsymbol{z}, \epsilon/L_i(r)\big),\ \text{s.t. } f\big(g_i(\boldsymbol{z}')\big) \neq f^*\big(g_i(\boldsymbol{z}')\big)\Big]}_{I} - \delta \qquad (4.5.1)\end{aligned}$$

where the first inequality is due to $\mu_i = (g_i)_*(\nu_d)$, and the second inequality holds because g_i is $L_i(r)$-locally Lipschitz with probability at least $1-\delta$ and $\mathcal{B}\big(\boldsymbol{z}, \epsilon/L_i(r)\big) \subseteq \mathcal{B}\big(\boldsymbol{z}, r\big)$ for any $\boldsymbol{z} \in \mathbb{R}^d$.

To further bound the term I, we make use of the Gaussian Isoperimetric Inequality as presented in Lemma 4.4.2. Let $\mathcal{A}_f = \{\boldsymbol{z} \in \mathbb{R}^d : f(g_i(\boldsymbol{z})) \neq f^*(g_i(\boldsymbol{z}))\}$ be the corresponding error region in the latent space. By Lemma 4.4.2, we have

$$I \geq \Phi\bigg(\Phi^{-1}\big(\nu_d(\mathcal{A}_f)\big) + \frac{\epsilon}{L_i(r)}\bigg) = \Phi\bigg(\Phi^{-1}\big(\mathrm{Risk}_{\mu_i}(f)\big) + \frac{\epsilon}{L_i(r)}\bigg). \qquad (4.5.2)$$

Finally, plugging (4.5.2) into (4.5.1), we complete the proof. □

4.5.2 Proof of Theorem 4.4.4

Proof. According to Definition 4.3.2 and Theorem 4.4.3, for any $f \in \mathcal{F}_\alpha$, we have

$$\begin{aligned}\mathrm{Rob}_\mu^\epsilon(\mathcal{F}_\alpha) &\leq 1 + \delta - \sum_{i=1}^K p_i \cdot \Phi\bigg(\Phi^{-1}\big(\mathrm{Risk}_{\mu_i}(f)\big) + \frac{\epsilon}{L_i(r)}\bigg) \\ &\leq 1 + \delta - \sum_{i=1}^K p_i \cdot \Phi\bigg(\Phi^{-1}\big(\mathrm{Risk}_{\mu_i}(f)\big) + \frac{\epsilon}{L_{\max}(r)}\bigg), \qquad (4.5.3)\end{aligned}$$

where the last inequality holds because $\Phi(\cdot)$ is monotonically increasing. For any $f \in \mathcal{F}_\alpha$, let $\mathcal{E} = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) \neq f^*(\boldsymbol{x})\}$ be the error region and $\alpha_i = \mu_i(\mathcal{E})$ be the measure of $\mathcal{E}$ under the i-th conditional distribution.

Thus, to obtain an upper bound on $\mathrm{Rob}_\mu^\epsilon(\mathcal{F}_\alpha)$ using (4.5.3), it remains to solve the following optimization problem:

$$\underset{\alpha_1,\ldots,\alpha_K\in[0,1]}{\text{minimize}} \sum_{i=1}^{K} p_i\cdot\Phi\bigg(\Phi^{-1}(\alpha_i)+\frac{\epsilon}{L_{\max}(r)}\bigg) \quad \text{subject to} \quad \sum_{i=1}^{K} p_i\alpha_i \geq \alpha. \tag{4.5.4}$$

Note that for classifier in $\widetilde{\mathcal{F}}_\alpha$, by definition, we can simply replace $\alpha_i = \alpha$ in (4.5.4), which proves the upper bound on $\mathrm{Rob}_\mu^\epsilon(\widetilde{\mathcal{F}}_\alpha)$.

Next, we are going to show that the optimal value of (4.5.4) is achieved, only if there exists a class $i'\in[K]$ such that $\alpha_{i'} = \alpha/p_{i'}$ and $\alpha_i = 0$ for any $i\neq i'$. Consider the simplest case where $K = 2$. Note that $\Phi(\cdot)$ and $\Phi^{-1}(\cdot)$ are both monotonically increasing functions, which implies that $\sum_{i=1}^{K} p_i\alpha_i = \alpha$ holds when optimum achieved, thus the optimization problem for $K = 2$ can be formulated as follows

$$\min_{\alpha_1,\alpha_2\in[0,1]} \; p_1\cdot\Phi\bigg(\Phi^{-1}(\alpha_1)+\frac{\epsilon}{L_{\max}(r)}\bigg)+p_2\cdot\Phi\bigg(\Phi^{-1}(\alpha_2)+\frac{\epsilon}{L_{\max}(r)}\bigg) \quad \text{s.t. } p_1\alpha_1+p_2\alpha_2=\alpha. \tag{4.5.5}$$

Suppose $\alpha_1 \geq \alpha_2$ holds for the initial setting. Now consider another setting where $\alpha_1' > \alpha_1$, $\alpha_2' < \alpha_2$. Let $s_1 = \Phi^{-1}(\alpha_1') - \Phi^{-1}(\alpha_1)$ and $s_2 = \Phi^{-1}(\alpha_2) - \Phi^{-1}(\alpha_2')$. According to the equality constraint of the optimization problem (4.5.5), we have

$$p_1\cdot\int_{\Phi^{-1}(\alpha_1)}^{\Phi^{-1}(\alpha_1)+s_1}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-x^2/2}dx = p_2\cdot\int_{\Phi^{-1}(\alpha_2)-s_2}^{\Phi^{-1}(\alpha_2)}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-x^2/2}dx. \tag{4.5.6}$$

Let $\eta = \epsilon/L_{\max}(r)$ for simplicity. By simple algebra, we have

$$\begin{aligned}
&p_1\cdot\int_{\Phi^{-1}(\alpha_1)+\eta}^{\Phi^{-1}(\alpha_1)+s_1+\eta}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-x^2/2}dx\\
&= p_1\cdot\int_{\Phi^{-1}(\alpha_1)}^{\Phi^{-1}(\alpha_1)+s_1}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-u^2/2-\eta\cdot u-\eta^2/2}du\\
&< p_1\cdot\exp^{-\eta\cdot\Phi^{-1}(\alpha_1)-\eta^2/2}\cdot\int_{\Phi^{-1}(\alpha_1)}^{\Phi^{-1}(\alpha_1)+s_1}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-u^2/2}du\\
&\leq p_2\cdot\exp^{-\eta\cdot\Phi^{-1}(\alpha_2)-\eta^2/2}\cdot\int_{\Phi^{-1}(\alpha_2)-s_2}^{\Phi^{-1}(\alpha_2)}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-u^2/2}du\\
&< p_2\cdot\int_{\Phi^{-1}(\alpha_2)-s_2+\eta}^{\Phi^{-1}(\alpha_2)+\eta}\frac{1}{\sqrt{2\pi}}\cdot\exp^{-x^2/2}dx,
\end{aligned}$$

where the first inequality holds because $\exp^{-\eta\cdot u} < \exp^{-\eta\cdot\Phi^{-1}(\alpha_1)}$ for any $u > \Phi^{-1}(\alpha_1)$, the second inequality follows from (4.5.6) and the fact that $\Phi^{-1}(\alpha_1) \geq \Phi^{-1}(\alpha_2)$, and the last inequality holds because $\exp^{-\eta\cdot\Phi^{-1}(\alpha_2)} < \exp^{-\eta\cdot u}$ for any $u < \Phi^{-1}(\alpha_2)$. Therefore, the optimal value of (4.5.5) will be achieved when $\alpha_1 = 0$ or $\alpha_2 = 0$. For general setting with $K > 2$, since $\alpha_1, \ldots, \alpha_K$ are independent in the objective, we can fix $\alpha_3, \ldots, \alpha_K$ and optimize α_1 and α_2 first, then deal with α_i incrementally using the same technique. $\square$

4.6 Experimental Details

This section provides additional details for our experiments.

4.6.1 Network Architectures and Hyper-parameter Settings

For the certified robust defense (LP-Certify), we adopt the the same four-layer neural network architecture as implemented in [WSMK18], with two convolutional layers and two fully connected layers, and use the an Adam optimizer with learning rate 0.001 and batch size 50 for training the robust classifier. In particular, the adversarial loss function is based on the robust certificate under ℓ_2 proposed in [WSMK18].

For training attack-based robust models (Adv-Train and TRADES), we use a seven-layer CNN architecture which contains four convolution layers and three fully connected layers. We use a SGD optimizer to minimize the attack-based adversarial loss with learning rate 0.05 on MNIST and learning rate 0.01 on ImageNet10. Table 4.1 summarizes all the hyper-parameters we used for training the robust models (β is an additional parameter specifically used in TRADES).

For evaluating the unconstrained adversarial robustness, we implemented PGD attack with ℓ_2 metric. Table 4.2 shows all the hyper-parameters we used for robustness evaluation.

Table 4.1: Hyper-parameters used for training robust models.

Para.	Generated MNIST			ImageNet10	
	LP-Certified	Adv Training	TRADES	Adv Training	TRADES
ϵ (in ℓ_2)	2.0	3.0	3.0	3.0	3.0
optimizer	ADAM	SGD	SGD	SGD	SGD
learning rate	0.001	0.05	0.05	0.01	0.01
#epochs	60	100	100	100	100
attack step size	-	0.5	0.5	0.5	0.5
#attack steps	-	40	40	10	10
β	-	-	6.0	-	6.0

Table 4.2: Hyper-parameters used for evaluating the model robustness via PGD attack.

Para.	Generated MNIST			ImageNet10		
	$\epsilon = 1.0$	$\epsilon = 2.0$	$\epsilon = 3.0$	$\epsilon = 1.0$	$\epsilon = 2.0$	$\epsilon = 3.0$
attack step size	0.1	0.3	0.5	0.1	0.3	0.5
#attack steps	100	100	100	100	100	100

4.6.2 Strategies for Estimating In-distribution Adversarial Robustness

Initialization of $\boldsymbol{z}$: For MNIST data, we design an initialization strategy for $\boldsymbol{z}$ in order to make sure the perturbation term $\|G(\boldsymbol{z}, y) - \boldsymbol{x}\|_2$ can be efficiently optimized. To be more specific, starting from random noise, we first solve another optimization problem:

$$\boldsymbol{z}_{\text{init}} = \operatorname*{argmin}_{\boldsymbol{z}} \|G(\boldsymbol{z}, y) - \boldsymbol{x}\|_2.$$

By setting $\boldsymbol{z}_{\text{init}}$ as our initial point, we minimize the initial perturbation distance. Here $\boldsymbol{z}$ can start from any random initial point as we will then optimize the generated image under ℓ_2 distance.

For ImageNet10 data, even applying the above optimization procedure doesn't result in an initial $\boldsymbol{z}$ such that $\|G(\boldsymbol{z}, y) - \boldsymbol{x}\|_2 \leq \epsilon$ when ϵ is small. Therefore, we use another strategy by recording the $\boldsymbol{z}^*$ when generating the test sample $\mathbf{x}$, i.e., $G(\boldsymbol{z}^*, y) = \mathbf{x}$. And we adopt $\boldsymbol{z}^*$ as the initial point for $\boldsymbol{z}$ in solving (4.7.2). This makes sure that the whole optimization procedure could at least find one point satisfying the perturbation constraint[1].

The choice of λ: Inspired by [CW17], we also adopt binary search strategy for finding better regularization parameter λ. Specifically, we set initial $\lambda = 1.0$ and if we successfully find an adversarial example, we lower the value of λ via binary search. Otherwise, we raise the value of λ. For each batch of examples, we perform 5 times binary search in order to find qualified in-distribution adversarial examples.

Hyper-parameters: We use Adam optimizer with learning rate 0.01 for finding in-distribution adversarial examples. We set maximum iterations for each λ binary search as 10000.

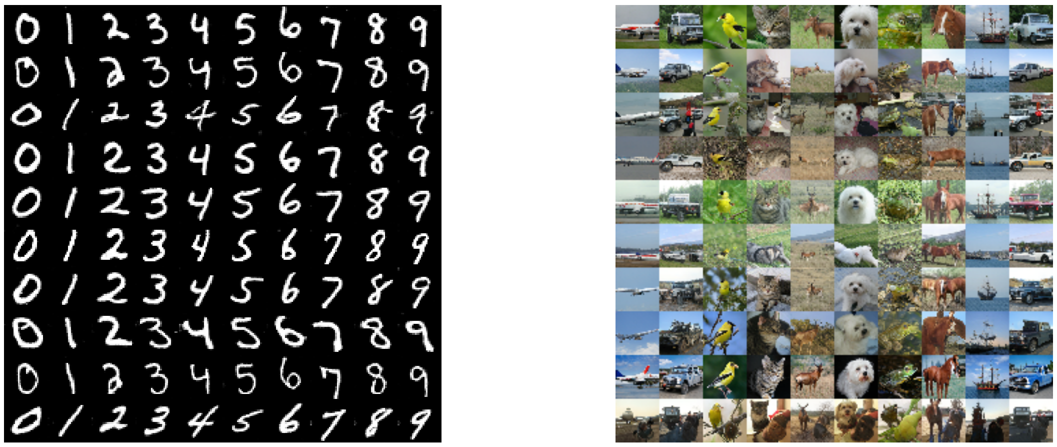
4.7 Experiments

This section provides our empirical evaluations of the intrinsic robustness on real image distributions to evaluate the tightness of our bound. We test our bound on two image distributions generated using MNIST [LBB+98] and ImageNet [DDS+09] datasets. Code for all our experiments is available at `https://github.com/xiaozhanguva/Intrinsic-Rob`.

4.7.1 Conditional GAN Models

Instead of directly evaluating the robustness on real datasets, we make use of conditional GAN models to generate datasets from the learned data distributions and evaluate the robustness of several state-of-the-art robust models trained on the generated dataset for a fair

[1]We didn't use $\boldsymbol{z}^*$ as the initialization for MNIST data as our empirical study shows that the optimization-based initialization achieves better performances on MNIST.



(a) ACGAN Generated MNIST (b) BigGAN Generated ImageNet

Figure 4.1: Illustration of the generated images using different conditional models. For BigGAN generated images, we select 10 specific classes from the 1000 ImageNet classes (corresponding to the 10 image classes in CIFAR-10).

comparison with the theoretical robustness limits. Note that this approach is only feasible with conditional generative models as unconditional models cannot provide the corresponding labels for the generated data samples. For MNIST, we adopt ACGAN [OOS17] which features an additional auxiliary classifier for better conditional image generation. The ACGAN model generates 28×28 images from a 100-dimension latent space concatenated with an addition 10-dimension one-hot encoding of the conditional class labels. For ImageNet, we adopt the BigGAN model [BDS19] which is the state-of-the-art GAN model in conditional image generation. It generates 128×128 images from a 120-dimension latent space. We down-sampled the generated images to 32×32 for efficiency propose. We consider a standard Gaussian[2] as the latent distribution for both conditional generative models. Figure 4.1 shows examples of the generated MNIST and ImageNet images. For both figures, each column of images corresponds to a particular label class of the considered dataset.

[2]The original BigGAN model uses truncated Gaussian. We adapted it to standard Gaussian distribution.

Table 4.3: Local Lipschitz constants of ACGAN model on MNIST classes with $r = 0.5$ and $\delta = 0.001$.

Class	digit 0	digit 1	digit 2	digit 3	digit 4	digit 5	digit 6	digit 7	digit 8	digit 9
$L(r)$	7.9	8.6	8.3	7.8	10.3	11.0	9.5	7.8	9.3	10.9

Table 4.4: Local Lipschitz constants of BigGAN on 10 selected ImageNet classes with $r = 0.5$ and $\delta = 0.001$.

Class	airliner	jeep	goldfinch	tabby cat	hartebeest	Maltese dog	bullfrog	sorrel	pirate ship	pickup
$L(r)$	13.1	14.5	11.7	12.4	10.4	11.3	9.4	13.0	13.1	14.9

4.7.2 Local Lipschitz Constant Estimation

From Theorem 4.4.4, we observe that given a class of classifiers with risk at least α, the derived intrinsic robustness upper bound is mainly decided by the perturbation strength ϵ and the local Lipschitz constant $L(r)$. While ϵ is usually predesignated in common robustness evaluation settings, the local Lipschitz constant $L(r)$ is unknown for most real world tasks. Computing an exact Lipschitz constant of a deep neural network is a difficult open problem. Thus, instead of obtaining the exact value, we approximate $L(r)$ using a sample-based approach with respect to the generative models.

Recalling Definition 4.4.1, we consider Δ as the ℓ_2 distance and $g(\boldsymbol{z})$ and $g(\boldsymbol{z}')$ are easy to compute via the generator network. Computing $L(r)$, however, is much more complicated as it requires obtaining a maximum value within a radius-r ball. To deal with this, our approach approximates $L(r)$ by sampling N points in the neighborhood around $\boldsymbol{z}$ and takes the maximum value as the estimation of the true maximum value within the ball. Since the definition of local Lipschitz is probabilistic, we take multiple samples of the latent vectors $\boldsymbol{z}$ to estimate the local Lipschitz constant $L(r)$. The estimation procedure is summarized in Algorithm 7, which gives an underestimate of the underlying truth. Developing better Lipschitz estimation methods is an active area in machine learning research, but is not the main focus of this work.

Algorithm 7 Local Lipschitz Estimation

```
**Input:** number of samples $S$, number of local neighbors per sample $N$, $r$, $\delta$
for $i = 1, \ldots, S$ do
    Generate a latent space sample $\boldsymbol{z}_i$
    Generate $N$ samples $\{\widehat{\boldsymbol{z}}_i^j\}_{j=1}^N$ within $\mathcal{B}_r(\boldsymbol{z}_i)$
    $L_i = \max_j \frac{\|g(\widehat{\boldsymbol{z}}_i^j) - g(\boldsymbol{z}_i)\|_2}{\|\widehat{\boldsymbol{z}}_i^j - \boldsymbol{z}_i\|_2}$
end for
**Output:** $(1-\delta)$-percentile of $\{L_i\}_{i=1}^S$
```

Tables 4.3 and 4.4 summarize the local Lipschitz constants estimated for the trained ACGAN and BigGAN generators conditioned on each class. For both conditional generators, we set $S = 1000$, $N = 2000$, $r = 0.5$ and $\delta = 0.001$ in Algorithm 7 for Lipschitz estimation. For BigGAN, the specifically selected 10 classes from ImageNet are reported in Table 4.4. In addition, the reported estimates are averaged over 10 repeated trials, where the standard deviation varies from 0.2 to 0.6 for ACGAN and varies from 0.3 to 1.1 for BigGAN.

Compared with unconditional generative models, conditional ones generate each class using a separate generator. Thus, the local Lipschitz constant of each class-conditioned generator is expected to be smaller than that of unconditional ones, as the within-class variation is usually much smaller than the between-class variation for a given classification dataset. For instance, we trained an unconditional GAN generator [GPAM$^+$14] on MNIST dataset, which yields an overall local Lipschitz constant of 27.01 from Algorithm 7 under the same parameter settings. If we plug in this estimated Lipschitz constant into the theoretical results in [FFF18a], the implied intrinsic robustness bound is in fact vacuous (above 1) with perturbations strength $\epsilon \leq 3.0$ in ℓ_2 distance.

4.7.3 Comparisons with Robust Classifiers

We compare our derived intrinsic robustness upper bound with the empirical adversarial robustness achieved by the current state-of-the-art defense methods under ℓ_2 perturbations. Specifically, we consider three robust training methods: *LP-Certify*: optimization-based certified robust defense [WSMK18]; *Adv-Train*: PGD attack based adversarial training [MMS+18]; and *TRADES*: adversarial training by accuracy and robustness trade-off [ZYJ+19]. We adopt these robust training methods to train robust classifiers over a set of generated training images and evaluate their robustness on the corresponding generated test set.

For MNIST, we use our trained ACGAN model to generate 10 classes of hand-written digits with $60,000$ training images and $10,000$ testing images. For ImageNet, we use the BigGAN model to generate 10 selected classes of images, which contains $50,000$ images for training set and $10,000$ images for test set. We refer to the 10-class BigGAN generated dataset as 'ImageNet10'. We set $\epsilon = 3.0$ for training robust models using Adv-Train and TRADES for both generated datasets, whereas we only train the LP-based certified robust classifier with $\epsilon = 2.0$ on generated MNIST data, as it is not able to scale with ImageNet10 as well as generated MNIST with larger ϵ (see Appendix 4.6.1 for all the selected hyper-parameters and network architectures).

A commonly-used method to evaluate the robustness of a given model is by performing carefully-designed adversarial attacks. Here we adopt the PGD attack [MMS+18], and report the robust accuracy (classification accuracy on inputs generated using the PGD attack) as the empirically measured model robustness. We test both the natural classification accuracy and the robustness of the aforementioned adversarially trained classifiers under ℓ_2 perturbations with perturbation strength ϵ selected from $\{1.0, 2.0, 3.0\}$. See Appendix 4.6.1 for PGD parameter settings.

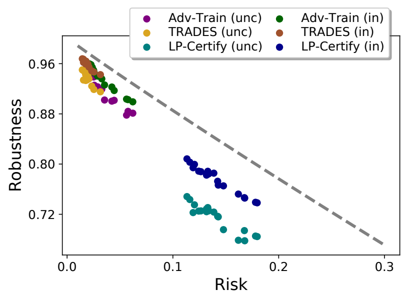
Table 4.5 compares the empirically measured robustness of the trained robust classifiers

Table 4.5: Comparisons between the empirically measured robustness of adversarially trained classifiers and the implied theoretical intrinsic robustness bound on the conditional generated datasets.

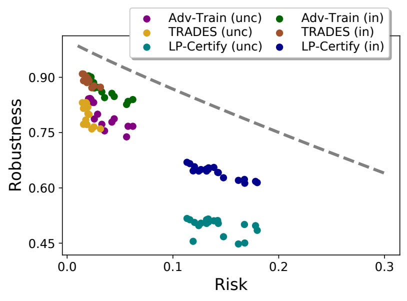
	ACGAN generated MNIST				BigGAN generated ImageNet10			
	$\epsilon = 0.0$	$\epsilon = 1.0$	$\epsilon = 2.0$	$\epsilon = 3.0$	$\epsilon = 0.0$	$\epsilon = 1.0$	$\epsilon = 2.0$	$\epsilon = 3.0$
LP-Certify	88.3%	$74.0 \pm 0.4\%$	$51.1 \pm 0.6\%$	$23.5 \pm 0.3\%$	-	-	-	-
Adv-Train	97.2%	$93.1 \pm 0.2\%$	$83.5 \pm 0.3\%$	$58.9 \pm 0.4\%$	82.1%	$67.8 \pm 0.3\%$	$47.1 \pm 0.4\%$	$23.4 \pm 0.4\%$
TRADES	98.3%	$94.8 \pm 0.2\%$	$81.8 \pm 0.4\%$	$57.7 \pm 0.4\%$	83.4%	$68.5 \pm 0.3\%$	$49.1 \pm 0.5\%$	$27.8 \pm 0.5\%$
Our Bound	-	98.2%	97.8%	97.2%	-	83.5%	81.8%	80.0%

and the derived theoretical upper bound on intrinsic robustness. More specifically, $\epsilon = 0$ corresponds to the standard classification. For empirically measured robust accuracy with $\epsilon > 0$, we report both the mean and the standard deviation over 10 repeated trials. For computing our theoretical robust bounds, we set the risk threshold $\alpha = 0.015$ for generated MNIST and $\alpha = 0.15$ for ImageNet10, to reflect the best natural accuracy achieved by the considered robust classifiers.

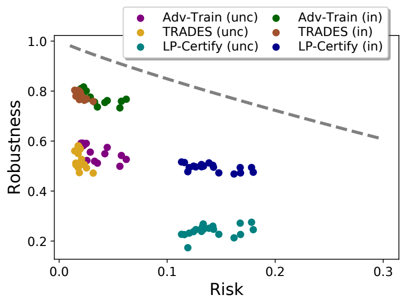
Under most settings, there exists a large gap between the robust limit implied by our theory and the best adversarial robustness achieved by state-of-the-art robust classifiers. For instance, Adv-Train and TRADES only achieve less than 50% robust accuracy on the generated ImageNet10 data with $\epsilon = 2.0$, whereas the estimated robustness bound is as high as 81.8%. The gap becomes even larger when we increase the perturbation strength ϵ. In contrast to the previous theoretical results on artificial distributions, for these image classification problems we cannot simply conclude from the intrinsic robustness bound that adversarial examples are inevitable. This huge gap between the empirical robustness of the best current image classifiers and the estimated theoretical bound suggests that either there is a way to train better robust models or that there exist other explanations for the inherent limitations of robust learning against adversarial examples.



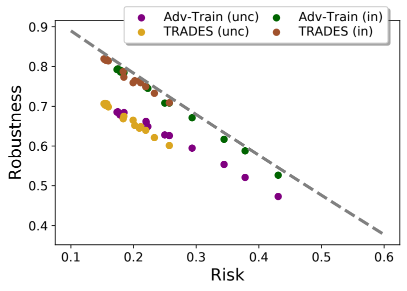
(a) Generated MNIST ($\epsilon = 1.0$)



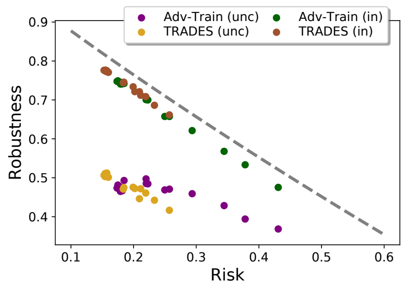
(b) Generated MNIST ($\epsilon = 2.0$)



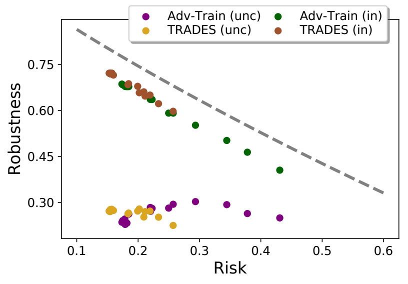
(c) Generated MNIST ($\epsilon = 3.0$)



(d) ImageNet10 ($\epsilon = 1.0$)



(e) ImageNet10 ($\epsilon = 2.0$)



(f) ImageNet10 ($\epsilon = 3.0$)

Figure 4.2: Comparisons between the theoretical intrinsic robustness bound and the empirically estimated unconstrained/in-distribution adversarial robustness, denoted as "unc" and "in" in the legend, of models produced during robust training on the generated data under ℓ_2. In each subfigure, the dotted curve line represents the theoretical bound on intrinsic robustness with horizontal axis denoting the different choice of α.

4.7.4 In-distribution Adversarial Robustness

In Section 4.7.3, we empirically show the unconstrained robustness of existing robust classifiers is far below the intrinsic robustness upper bound implied by our theory for real distributions. However, it is not clear whether the reason is that current robust training methods are far from perfect, or that our derived upper bound is not tight enough due to the Lipschitz relaxation step used for proving such bound. In this section, we empirically study the in-distribution adversarial risk for a better characterization of the actual intrinsic robustness. As shown in Remark 4.4.6, the in-distribution adversarial robustness of any classifier with risk at least α can be regarded as a lower bound for the intrinsic robustness $\mathrm{Rob}_{\mu}^{\epsilon}(\mathcal{F}_{\alpha})$. This provides us a more accurate characterization of the intrinsic robustness bound and enables better understanding of intrinsic robustness.

While there are many types of attack algorithms in the literature that can be used to evaluate the unconstrained robustness of a given classifier in the image space, little has been done in terms of how to evaluate the in-distribution robustness. In order to empirically evaluate the in-distribution robustness, we straightforwardly formulate the following optimization problem to find adversarial examples on the image manifold:

$$\min_{\boldsymbol{z}} \ \mathcal{L}(f(G(\boldsymbol{z}, y)), y) \ \text{ s.t. } \|G(\boldsymbol{z}, y) - \boldsymbol{x}\|_2 \leq \epsilon, \tag{4.7.1}$$

where $\boldsymbol{z} \in \mathbb{R}^d$, $\boldsymbol{x}$ is the data sample in the image space to be attacked, f is the given classifier, and $\mathcal{L}$ denotes the adversarial loss function. The goal of (4.7.1) is to optimize the latent vector to lower the adversarial loss (make the robust classifier mis-classify some generated images) while keeping the distance between the generated image and the test image within ϵ perturbation limit. The key difficulty in solving (4.7.1) lies in the fact that we cannot perform any type of projection operations as we are optimizing over $\boldsymbol{z}$ but the constraints are imposed on the generated image space $G(\boldsymbol{z}, y)$. This prohibits the use of common attack algorithms such as PGD. In order to solve (4.7.1), we transform (4.7.1) into the following Lagrangian formulation:

$$\min_{\boldsymbol{z}} \|G(\boldsymbol{z}, y) - \boldsymbol{x}\|_2 + \lambda \cdot \mathcal{L}(f(G(\boldsymbol{z}, y)), y). \tag{4.7.2}$$

This formulation ignores the perturbation constraint of ϵ and tries to find the in-distribution adversarial examples with the smallest possible perturbation. In order to evaluate the intrinsic robustness under a given ϵ perturbation budget, we need to further check all in-distribution adversarial examples found and only count those with perturbations within the ϵ constraint. Note that even though (4.7.2) provides us a feasible way to compute the in-distribution robustness of a classifier, equation (4.7.2) itself could be hard to solve in general.

First, it is not obvious how to initialize $\boldsymbol{z}$. Random initialization of $\boldsymbol{z}$ could lead to bad local optima which prevent the optimizer from efficiently solving (4.7.2) or even finding a $\boldsymbol{z}$ that could make $G(\boldsymbol{z}, y)$ close enough to $\boldsymbol{x}$. Second, the hyper-parameter λ could be quite sensitive to different test examples. Failing to choose a proper λ could also lead to failures in finding in-distribution adversarial examples within ϵ constraint. In order to the tackle the aforementioned challenges, we propose to solve another optimization problem for the initialization of $\boldsymbol{z}$ and adopt binary search for the best choice of λ (see Appendix 4.6.2 for more details of our implementation).

Figure 4.2 summarizes results from our empirical evaluations on intrinsic robustness of the generated MNIST and ImageNet10 data. We evaluate the empirical robustness of three types of robust training methods at different time points during the training procedure. To be more specific, we evaluate the robustness of the intermediate models produced every 5 training epochs. For each method, we plot both the unconstrained robustness measured by PGD attacks and the in-distribution robustness measured using the aforementioned strategies. In addition, based on the local Lipschitz constants estimated in Section 4.7.2, we plot the implied theoretical bound on intrinsic robustness as the dotted line curve for direct comparison.

Compared with the intrinsic robustness upper bound (dotted curve line), the unconstrained robustness of various robustly-trained models is much smaller, and the gap between them becomes more obvious as we increase ϵ. This aligns with our observations in Section 4.7.3. However under all the considered settings, the estimated in-distribution adversarial robustness is much higher than the unconstrained one and closer to the theoretical upper bound, especially for the ImageNet10 data. Note that according to Remark 4.4.6, the actual intrinsic robustness $\mathrm{Rob}_{\mu}^{\epsilon}(\mathcal{F}_{\alpha})$ should lie between the in-distribution robustness of any given classifier with risk at least α and the derived intrinsic robustness upper bound. Observing the big gap between the estimated in-distribution and unconstrained robustness of various robustly trained models, one would expect the current state-of-the-art robust models are still far from approaching the actual intrinsic robustness limit for real image distributions.

4.8 Conclusions

We studied the intrinsic robustness of typical image distributions using conditional generative models. By deriving theoretical upper bounds on intrinsic robustness and providing empirical estimates on the generated image distributions, we observed a large gap between the theoretical intrinsic robust limit and the best robustness achieved by state-of-the-art robust classifiers. Our results imply that the inevitability of adversarial examples claimed in recent theoretical studies, such as [FFF18a], do not apply to real image distributions, and suggest that there is a need for deeper understanding on the intrinsic robustness limitations for real data distributions.

CHAPTER 5

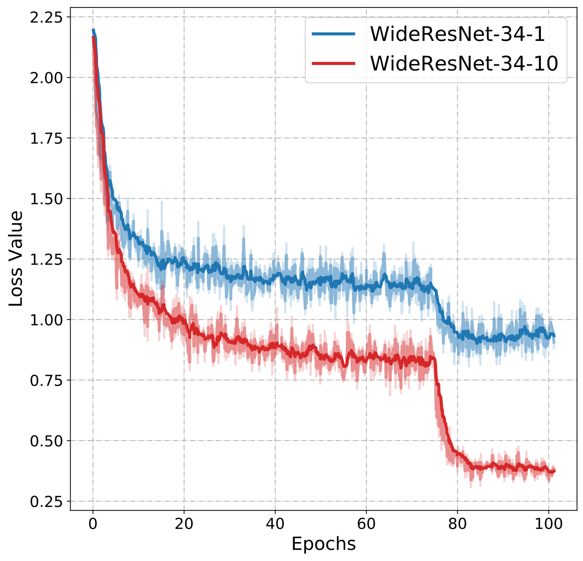
Do Wider Neural Networks Really Help Adversarial Robustness?

5.1 Introduction

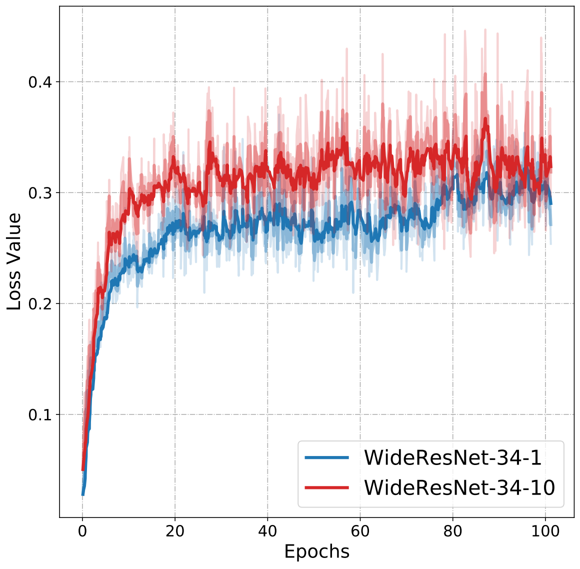
Researchers have found that Deep Neural Networks (DNNs) suffer badly from adversarial examples [SZS$^+$14]. By perturbing the original inputs with an intentionally computed, undetectable noise, one can deceive DNNs and even arbitrarily modify their predictions on purpose. To defend against adversarial examples and further improve model robustness, various defense approaches have been proposed [PMW$^+$16, MC17, DAL$^+$18, LLD$^+$18, XWZ$^+$18, GRCVDM18, SKN$^+$18, SKC18]. Among them, adversarial training [GSS15, MMS$^+$18] has been shown to be the most effective type of defenses [ACW18]. Adversarial training can be seen as a form of data augmentation by first finding the adversarial examples and then training DNN models on those examples. Specifically, given a DNN classifier f parameterized by $\boldsymbol{\theta}$, a general form of adversarial training with loss function $\mathcal{L}$ can be defined as:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\Big[\underbrace{\mathcal{L}(\boldsymbol{\theta};\mathbf{x}_i,y_i)}_{\text{natural risk}} + \lambda\cdot \underbrace{\max_{\widehat{\mathbf{x}}_i\in\mathbb{B}(\mathbf{x}_i,\epsilon)}\big[\mathcal{L}(\boldsymbol{\theta};\widehat{\mathbf{x}}_i,y_i)-\mathcal{L}(\boldsymbol{\theta};\mathbf{x}_i,y_i)\big]}_{\text{robust regularization}}\Big], \tag{5.1.1}$$

where $\{(\mathbf{x}_i, y_i)_{i=1}^{n}\}$ are training data, $\mathbb{B}(\mathbf{x},\epsilon) = \{\widehat{\mathbf{x}} \mid \|\widehat{\mathbf{x}}-\mathbf{x}\|_p \le \epsilon\}$ denotes the ℓ_p norm ball with radius ϵ centered at $\mathbf{x}$, and $p \ge 1$, and $\lambda > 0$ is the regularization parameter. Compared with standard empirical risk minimization, the extra robust regularization term encourages the data points within $\mathbb{B}(\mathbf{x},\epsilon)$ to be classified as the same class, i.e., encourages the predictions to be stable. The regularization parameter λ adjusts the strength of robust regularization. When $\lambda = 1$, it recovers the formulation in [MMS+18], and when $\lambda = 0.5$, it recovers the formulation in [GSS15]. Furthermore, replacing the loss difference in robust regularization term with the KL-divergence based regularization recovers the formulation in [ZYJ+19].



(a) Natural Risk



(b) Robust Regularization

Figure 5.1: Plots of both natural risk and robust regularization in (5.1.1). Two 34-layer WideResNet [ZK16] are trained by TRADES [ZYJ+19] on CIFAR10 [KH+09] with widen factor being 1 and 10.

One common belief in the practice of adversarial training is that, compared with the standard empirical risk minimization, adversarial training requires much wider neural networks to achieve better robustness. [MMS+18] provided an intuitive explanation: robust classification requires a much more complicated decision boundary, as it needs to handle the presence of possible adversarial examples. However, it remains elusive how does the network width affect model robustness. To answer this question, we first examine whether the larger network width contributes to both the natural risk term and the robust regularization term in (5.1.1). Interestingly, when tracing the value changes in (5.1.1) during adversarial training, we observe that the value of the robust regularization part actually gets worse on wider models, suggesting that larger network width does not lead to better stability in predictions. In Figure 5.1, we show the loss value comparison of two different wide models trained by TRADES [ZYJ+19] with $\lambda = 6$ as suggested in the original paper. We can see that the wider model (i.e., WideResNet-34-10) achieves better natural risk but incurs a larger value on robust regularization. This motivates us to find out the cause of this phenomenon.

In this paper, we study the relationship between neural network width and model robustness for adversarially trained neural networks. Our contributions can be summarized as follows:

1. We show that the model robustness is closely related to both natural accuracy and perturbation stability, a new metric we proposed to characterize the strength of robust regularization. The balance between the two is controlled by the robust regularization parameter λ. With the same value of λ, the natural accuracy is improved on wider models while the perturbation stability often worsens, leading to a possible decrease in the overall model robustness. This suggests that proper tuning of λ on wide models is necessary despite being extremely time-consuming, while directly using the fine-tuned λ on small networks to train wider ones, as many people did in practice [MMS+18, ZYJ+19], may lead to deteriorated model robustness.

2. Unlike previous understandings that there exists a trade-off between natural accuracy and robust accuracy, we show that the real trade-off should between natural accuracy and perturbation stability. And the robust accuracy is actually the consequence of this trade-off.

3. To understand the origin of the lower perturbation stability of wider networks, we further relate perturbation stability with the network's local Lipschitznesss. By leveraging recent results on neural tangent kernels [JHG18, AZLS19, ZCZG20, CG19, GCL+19], we show that with the same value of λ, larger network width naturally leads to worse perturbation

stability, which explains our empirical findings.

4. Our analyses suggest that to unleash the potential of wider model architectures fully, one should mitigate the perturbation stability deterioration and enlarge robust regularization parameter λ for training wider models. Empirical results verified the effectiveness of this strategy on benchmark datasets. In order to alleviate the heavy burden for tuning λ on wide models, we develop the Width Adjusted Regularization (WAR) method to transfer the knowledge we gain from fine-tuning smaller networks into the training of wider networks and significantly save the tuning time.

Notation. For a d-dimensional vector $\mathbf{x} = [x_1, ..., x_d]^\top$, we use $\|\mathbf{x}\|_p = (\sum_{i=1}^{d} |x_i|^p)^{1/p}$ with $p \geq 1$ to denote its ℓ_p norm. $\mathbb{1}(\cdot)$ represents the indicator function and $\forall$ represents the universal quantifier.

5.2 Related Work

Adversarial attacks: Adversarial samples and their intriguing properties were first found in [SZS+14]. Since then, tremendous works have been done exploring the origins of this intriguing property of deep learning [GR15, KGB17, FFF18b, TPG+17, GMF+18, ZCGE20] as well as designing more powerful attacks [GSS15, PMJ+16, MFF16, MMS+18, CW17, CG20] under various attack settings. [ACW18] identified the gradient masking problem and showed that many defense methods could be broken with a few changes on the attacker. [CZS+17] proposed gradient-free black-box attacks and [IEA+18, IEM19, CZYG20] further improved its efficiency. Recently, [IST+19, JBZB19] pointed out adversarial examples are generated from the non-robust or invariant features hidden in the training data.

Defensive adversarial learning: Many defense approaches have been proposed aiming to directly learn a robust model that is able to defend against adversarial attacks. [MMS+18] proposed a general framework of robust training by solving a min-max optimization problem. [WMB+19] proposed a new criterion to quantitatively evaluate the convergence quality.

[ZYJ$^+$19] theoretically studied the trade-off between natural accuracy and robust accuracy for adversarially trained models. [WZY$^+$20] followed this framework and further improved its robustness by differentiating correctly classified and misclassified examples. [CBG$^+$17] solve the problem by restricting the variation of outputs with respect to the inputs. [CRK19, SLR$^+$19, LAG$^+$19] developed provably robust adversarial learning methods that have the theoretical guarantees on robustness. Recent works in [WRK20, QMG$^+$19] focus on creating adversarial robust networks with faster training protocol. Another line of works focuses on increasing the effective size of the training data, either by pre-trained models [HLM19] or by semi-supervised learning methods [CRS$^+$19, AUH$^+$19, NMKM19]. Very recently, [WXW20] proposed to further conduct adversarial weight perturbation aside from input perturbation to obtain more robust models. [GQU$^+$20] achieves further robust models by practical techniques like stochastic weight averaging.

Robustness and generalization: Earlier works like [GSS15] found that adversarial learning can reduce overfitting and help generalization. However, as the arm race between attackers and defenses keeps going, it is observed that strong adversarial attacks can cause severe damage to the model's natural accuracy [MMS$^+$18, ZYJ$^+$19]. Many works [ZYJ$^+$19, TSE$^+$19, RXY$^+$19] attempt to explain this trade-off between robustness and natural generalization, while some other works proposed different perspectives. [SST$^+$18] confirmed that more training data has the potential to close this gap. [BLPR19] suggested that a robust model is computationally difficult to learn and optimize. [ZCGE20] showed that there is still a large gap between the currently achieved model robustness and the theoretically achievable robustness limit on real image distributions. Very recently, [RXY$^+$20] showed that this tradeoff stems from overparameterization and insufficient data in the linear regression setting. [YRZ$^+$20] proved that both accuracy and robustness are achievable through locally Lipschitz functions with separated data and the gap between theory and practice is due to either failure to impose local Lipschitzness or insufficient generalization.

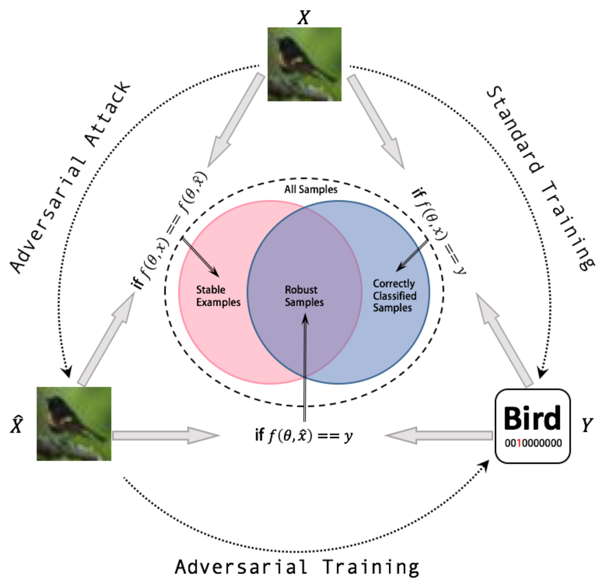


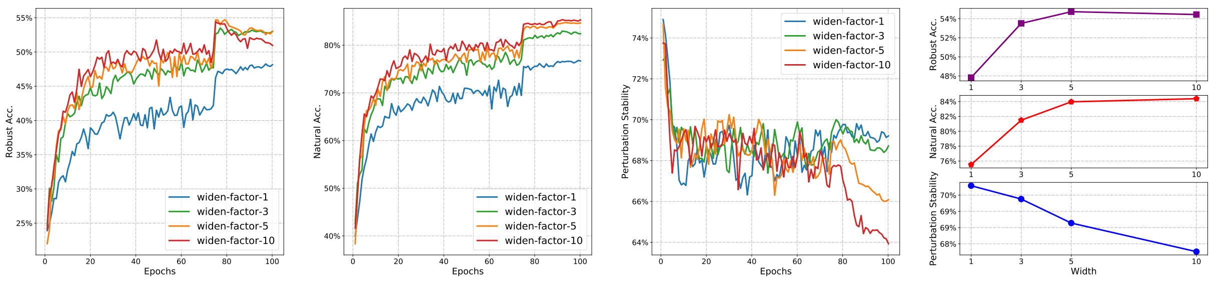
Figure 5.2: An illustration of the robust samples, correctly classified samples, and stable samples in (5.3.1).

5.3 Empirical Study on Network Width and Adversarial Robustness

In this section, we empirically study the relation between network width and robustness in a more thorough way by first taking a closer look at the robust accuracy and the associated robust examples.

5.3.1 Characterization of Robust Examples

Robust accuracy is the standard evaluation metric of robustness, which measures the ratio of robust examples, i.e., examples that can still be correctly classified after adversarial attacks. Previous empirical results suggest that wide models enjoy both better generalization ability and model robustness. Specifically, [MMS$^+$18] proposed to extend ResNet [HZRS16b] architecture to WideResNet [ZK16] with a widen factor 10 for adversarial training on the CIFAR10 dataset and found that the increased model capacity significantly improves both



(a) Robust Accuracy (b) Natural Accuracy (c) Perturbation Stability (d) Metrics vs. Width

Figure 5.3: Plots of (a) robust accuracy, (b) natural accuracy, and (c) perturbation stability against training epochs for networks of different width. Results are acquired on CIFAR10 with the adversarial training method TRADES and architectures of WideResNet-34. Training schedule is the same as the original work [ZYJ+19]. We record all three metrics when robust accuracy reaches the highest point and plot them against network width in (d).

robust accuracy and natural accuracy. Later works such as [ZYJ+19, WZY+20] follow this finding and report their best result using WideResNet [ZK16] with widen factor 10.

However, as shown by our findings in Figure 5.1, wider models actually lead to worse robust regularization effects, suggesting that wider models are not better in all aspects and the relation between model robustness and network width may be more intricate than what people understood previously. To understand the intrinsic relationship between model robustness and network width, let us first take a closer look at the robust examples. Mathematically, robust examples can be defined as $\mathcal{S}_{\text{rob}} := \big\{\mathbf{x} : \forall \widehat{\mathbf{x}} \in \mathbb{B}(\mathbf{x}, \epsilon), f(\boldsymbol{\theta}; \widehat{\mathbf{x}}) = y\big\}$. Note that by definition of robust examples, we have the following equation holds:

$$\underbrace{\big\{\mathbf{x} : \forall \widehat{\mathbf{x}} \in \mathbb{B}(\mathbf{x}, \epsilon), f(\boldsymbol{\theta}; \widehat{\mathbf{x}}) = y\big\}}_{\text{robust examples:}\mathcal{S}_{\text{rob}}} = \underbrace{\big\{\mathbf{x} : f(\boldsymbol{\theta}; \mathbf{x}) = y\big\}}_{\text{correctly classified examples:}\mathcal{S}_{\text{correct}}}$$
$$\wedge \underbrace{\big\{\mathbf{x} : \forall \widehat{\mathbf{x}} \in \mathbb{B}(\mathbf{x}, \epsilon), f(\boldsymbol{\theta}; \mathbf{x}) = f(\boldsymbol{\theta}; \widehat{\mathbf{x}})\big\}}_{\text{stable examples:}\mathcal{S}_{\text{stable}}}, \tag{5.3.1}$$

where $\wedge$ is the logical conjunction operator. (5.3.1) suggests that the robust examples are the intersection of two other sets: the correctly classified examples (examples whose predictions are the correct labels) and the stable examples (examples whose predictions are the same within the ℓ_p norm ball). A more direct illustration of this relationship can be found in Figure 5.2. While the natural accuracy measures the ratio of correctly classified examples $|\mathcal{S}_{\text{correct}}|$ against the whole sample set, to our knowledge, there does not exist a metric measuring the ratio of stable examples $|\mathcal{S}_{\text{stable}}|$ against whole the sample set. Here we formally define this ratio as the *perturbation stability*, which measures the fraction of examples whose predictions cannot be perturbed as reflected in the robust regularization term in (5.1.1).

5.3.2 Evaluation of Perturbation Stability

We apply the TRADES [ZYJ+19] method, which is one of the strongest baselines in robust training, on CIFAR10 dataset and plot the robust accuracy, natural accuracy, and perturbation stability against the training epochs in Figure 5.3. Experiments are conducted on WideResNet-34 [ZK16] with various widen factors. For each network, when robust accuracy reaches the highest point, we record all three metrics and show their changing trend against network width in Figure 5.3(d). From Figure 5.3(d), we can observe that the perturbation stability decreases monotonically as the network width increases. This suggests that wider models are actually more vulnerable to adversarial perturbation. In this sense, the increased network width could hurt the overall model robustness to a certain extent. This can be seen from Figure 5.3(d), where the robust accuracy of widen-factor 5 is actually slightly better than that of widen-factor 10.

Aside from the relation with model width, we also gain other insights from the newly proposed perturbation stability:

1. Unlike robust accuracy and natural accuracy, perturbation stability gradually gets worse during the training process. This makes sense since an unlearned model that always outputs the same label will have perfect stability, and the training process tends to break

this perfect stability. From another perspective, the role of robust regularization in (5.1.1) is to encourage perturbation stability, such that the learned models cannot be easily perturbed for the sake of model robustness.

2. Previous works [ZYJ+19, TSE+19, RXY+19] have argued that there exists a trade-off between natural accuracy and robust accuracy. However, from (5.3.1), we can see that robust accuracy and natural accuracy are coupled with each other, as a robust example must first be correctly classified. When the natural accuracy goes to zero, the robust accuracy will become zero. On the other hand, higher natural accuracy also implies that more examples will likely become robust examples. Works including [RXY+20] and [Nak19] also challenged this robust-natural trade-off [TSE+19] does not hold for some cases. Therefore, we argue that the real trade-off here should be between natural accuracy and perturbation stability. And the robust accuracy is actually the consequence of this trade-off.

3. [RWK20a] has recently shown that adversarial training suffers from over-fitting as the robust accuracy might get worse as training proceeds, which can be seen in Figure 5.3(a). We found that the origin of this over-fitting is mainly attributed to the degenerate perturbation stability (Figure 5.3(c)) rather than the natural risk (Figure 5.3(b)). Future works of adversarial training may consider evaluating our perturbation stability to understand how their method takes effects. Do they only help natural risk, or robust regularization, or maybe both of them.

5.4 Why Larger Network Width Leads to Worse Perturbation Stability?

Our empirical findings in Section 5.3 explains why the larger network width may not help model robustness as it leads to worse perturbation stability. However, it still remains unclear what is the underlying reasons for the negative correlation between the perturbation stability and the model width. In this section, we show that larger network width naturally leads to worse perturbation stability from a theoretical perspective. Specifically, we first relate perturbation stability with the network's local Lipschitzness and then study the relationship between local Lipschitzness and the model width by leveraging recent studies on neural tangent kernels [JHG18, AZLS19, CG19, ZCZG20, GCL+19].

5.4.1 Perturbation Stability and Local Lipschitzness

Previous works [HA17, WZC+18] usually relate local Lipschitzness with network robustness, suggesting that smaller local Lipschitzness leads to robust models. Here we show that local Lipshctzness is more directly linked to perturbation stability, through which it further influences model robustness.

To get started, let us first recall the definition of Lipschitz continuity and its relation with gradient norms.

Lemma 5.4.1 (Lipschitz continuity and gradient norm [PŽ06])**.** Let $\mathcal{D} \in \mathbb{R}^d$ denotes a convex compact set, f is a Lipschitz function if for all $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$, it satisfies

$$|f(\mathbf{x}') - f(\mathbf{x})| \leq L\|\mathbf{x}' - \mathbf{x}\|_p,$$

where $L = \sup_{\mathbf{x}\in\mathcal{D}}\{\|\nabla f(\mathbf{x})\|_q\}$ and $1/p + 1/q = 1$.

Intuitively speaking, Lipschitz continuity guarantees that small perturbation in the input will not lead to large changes in the function output. In the adversarial training setting where the perturbation $\mathbf{x}'$ can only be chosen within the neighborhood of $\mathbf{x}$, we focus on the local Lipschitz constant where we restrict $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, \epsilon)$ and $L = \sup_{\mathbf{x}'\in\mathbb{B}(\mathbf{x},\epsilon)}\{\|\nabla f(\mathbf{x}')\|_q\}$.

Now suppose our neural network loss function is local Lipschitz, let $\mathbf{x}'$ be our computed adversarial example $\widehat{\mathbf{x}}$ and $\mathbf{x}$ be the original example, the robust regularization term satisfies

$$\max_{\widehat{\mathbf{x}}\in\mathbb{B}(\mathbf{x},\epsilon)} \big[\mathcal{L}(\boldsymbol{\theta}; \widehat{\mathbf{x}}, y) - \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y)\big] \leq L \max_{\widehat{\mathbf{x}}\in\mathbb{B}(\mathbf{x},\epsilon)} \big[\|\widehat{\mathbf{x}} - \mathbf{x}\|_p\big] \leq \epsilon L, \tag{5.4.1}$$

where the first inequality is due to local Lipschitz continuity and $L = \sup_{\mathbf{x}' \in \mathbb{B}(\mathbf{x},\epsilon)}\{\|\nabla \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}', y)\|_q\}$. (5.4.1) shows that the local Lipschitz constant is directly related to the robust regularization term, which can be used as a surrogate loss for the perturbation stability.

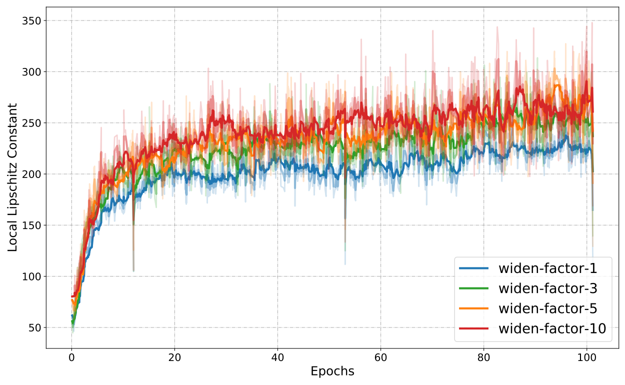


Figure 5.4: Plot of approximated local Lipschitz constant along the adversarial training trajectory. Models are trained by TRADES [ZYJ+19] on CIFAR10 dataset using WideResNet model. Wider networks in general have larger local Lipschitz constants.

5.4.2 Local Lipschitzness and Network Width

Now we study how the network width affects the perturbation stability via studying the local Lipschitz constant.

Recently, a line of research emerges, which tries to theoretically understand the optimization and generalization behaviors of over-parameterized deep neural networks through the lens of the neural tangent kernel (NTK) [JHG18, AZLS19, CG19, ZCZG20]. By showing the equivalence between over-parameterized neural networks and NTK in the finite width setting, this type of analysis characterizes the optimization and generalization performance of deep learning by the network architecture (e.g., network width, which we are particularly interested in). Recently, [GCL+19] also analyzed the convergence of adversarial training for over-parameterized neural networks using NTK. Here, we will show that the local Lipschitz constant increases as the model width.

In specific, let m be the network width and H be the network depth. Define an H-layer fully connected neural network as follows

$$f(\mathbf{x}) = \mathbf{a}^\top \sigma(\mathbf{W}^{(H)}\sigma(\mathbf{W}^{(H-1)} \cdots \sigma(\mathbf{W}^{(1)}\mathbf{x}) \cdots)),$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{m\times d}$, $\mathbf{W}^{(h)} \in \mathbb{R}^{m\times m}, h = 2, \ldots, H$ are the weight matrices, $\mathbf{a} \in \mathbb{R}^m$ is the output layer weight vector, and $\sigma(\cdot)$ is the entry-wise ReLU activation function. For notational simplicity, we denote by $\mathbf{W} = \{\mathbf{W}^{(H)}, \ldots, \mathbf{W}^{(1)}\}$ the collection of weight matrices and by $\mathbf{W}_0 = \{\mathbf{W}_0^{(H)}, \ldots, \mathbf{W}_0^{(1)}\}$ the collection of initial weight matrices. Following [GCL+19], we assume the first layer and the last layer's weights are fixed, and $\mathbf{W}$ is updated via projected gradient descent with projection set $B(R) = \{\mathbf{W} : \|\mathbf{W}^{(h)} - \mathbf{W}_0^{(h)}\|_F \le R/\sqrt{m}, h = 1, 2, \ldots, H\}$. We have the following lemma upper bounding the input gradient norm.

Lemma 5.4.2. For any given input $\mathbf{x} \in \mathbb{R}^d$ and ℓ_2 norm perturbation limit ϵ, if $m \ge \max(d, \Omega(H\log(H)))$, $R/\sqrt{m}+\epsilon \le c/(H^6(\log m)^3)$ for some sufficient small $c > 0$, then with probability at least $1 - O(H)e^{-\Omega(m(R/\sqrt{m}+\epsilon)^{2/3}H)}$, we have for any $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, \epsilon)$ and Lipschitz loss $\mathcal{L}$, the input gradient norm satisfies

$$\|\nabla \mathcal{L}(f(\mathbf{x}'), y)\|_2 = O\big(\sqrt{mH}\big).$$

The proof of Lemma 5.4.2 can be found in the supplemental materials. Note that Lemma 5.4.2 holds for any $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, \epsilon)$, therefore, the maximum input gradient norm in the ϵ-ball is also in the order of $O(\sqrt{mH})$. Lemma 5.4.2 suggests that the local Lipschitz constant is closely related to the neural network width m. In particular, the local Lipschitz constant scales as the square root of the network width. This in theory explains why wider networks are more vulnerable to adversarial perturbation.

In order to further verify the above theoretical result, we empirically calculate the local Lipschitz constant. In detail, for commonly used ℓ_∞ norm threat model, we evaluate the quantity $\sup_{\mathbf{x}'\in\mathbb{B}(\mathbf{x},\epsilon)}\{\|\nabla \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}', y)\|_1\}$ along the adversarial training trajectory for networks with different widths. Note that solving this maximization problem along the entire training trajectory is computationally expensive or even intractable. Therefore, we approximate this quantity by choosing the maximum input gradient ℓ_1-norm among the 10 attack steps for each iteration. We plot this result in Figure 5.4 and we can see that larger network width indeed leads to larger local Lipschitz constant values. This backup the theoretical results in Lemma 5.4.2.

5.5 Experiments

From Section 5.4, we know that wider networks have worse perturbation stability. This suggests that to fully unleash the potential of wide model architectures, we need to carefully control the decreasing of the perturbation stability on wide models. One natural strategy to do this is by adopting a larger robust regularization parameter λ in (5.1.1). In this section, we conduct thorough experiments to verify whether this strategy can mitigate the negative effects on perturbation stability and achieve better performances for wider networks.

It is worth noting that due to the high computational overhead of adversarial training on wide networks, previous works [ZYJ+19] tuned λ on smaller networks (ResNet18 [HZRS16a]) and directly apply it on wider ones, neglecting the influence of model capacity. Our analysis suggests that using the same λ for models with different widths is suboptimal, and one should use a larger λ for wider models in order to get better model robustness.

5.5.1 Experimental Settings

We conduct our experiments on CIFAR10 [KH+09] dataset, which is the most popular dataset in the adversarial training literature. It contains images from 10 different categories, with $50k$ images for training and $10k$ for testing. Here we first conduct our experiments using the TRADES [ZYJ+19] method. Networks are chosen from WideResNet [ZK16] with different widen factor from $1, 5, 10$. The batch size is set to 128, and we train each model for

100 epochs. The initial learning rate is set to be 0.1. We adopt a slightly different learning rate decay schedule: instead of dividing the learning rate by 10 after 75-th epoch and 90-th epoch in [MMS+18, ZYJ+19, WZY+20], we halve the learning rate for every epoch after the 75-th epoch, for the purpose of preventing over-fitting. For evaluating the model robustness, we perform the standard PGD attack [MMS+18] using 20 steps with step size 0.007, and $\epsilon = 8/255$. Note that previous works [ZYJ+19, WZY+20] report their results using step size 0.003, which we found is actually less effective than ours. All experiments are conducted on a single NVIDIA V100 GPU.

5.5.2 Model Robustness with Larger Robust Regularization Parameter

We first compare the robustness performance of models with different network width using robust regularization parameters chosen from $\{6, 9, 12, 15, 18, 21\}$ for TRADES [ZYJ+19]. Results of different evaluation metrics are presented in Table 5.1.

From Table 5.1, we can observe that the best robust accuracy for width-1 network is achieved when $\lambda = 9$, yet for width-5 network, the best robust accuracy is achieved when $\lambda = 12$, and for width-10 network, the best λ is 18. This suggests that wider networks indeed need a larger robust regularization parameter to unleash the power of wide model architecture fully. Our exploration also suggests that the optimal choice of λ for width-10 network is 18 under the same setting as [ZYJ+19], which is three times larger than the one used in the original paper, leading to an average improvement of 2.25% on robust accuracy. It is also worth noting that enlarging λ indeed leads to improved perturbation stability. Under the same λ, wider networks have worse perturbation stability. This observation is rather consistent with our empirical and theoretical findings in Sections 5.3 and 5.4. As stated in Section 5.3.2, the real trade-off is between natural accuracy and perturbation stability rather than robust accuracy. Also, the stability provides a clear hint for finding the best choice of λ.

We further show that our strategy also applies to the original adversarial training [MMS+18],

Table 5.1: The three metrics under PGD attack with different λ on CIFAR10 dataset using WideResNet-34 model. We test TRADES as well as our (generalized) adversarial training. Each experiment is repeated three times. The highest robustness value for each column is annotated with bold number. From the table, we can tell that: 1) The best choice of λ increases as the network width increases; 2) For models with the same width, the larger λ always leads to higher perturbation stability; 3) With the same λ, the larger width always hurts perturbation stability, which backs up our claim in Section 5.4.2.

	Robust Accuracy (%)			Natural Accuracy (%)			Perturbation Stability (%)		
λ	width-1	width-5	width-10	width-1	width-5	width-10	width-1	width-5	width-10
TRADES [ZYJ+19]									
6	47.81±.09	54.45±.16	54.18±.39	**76.26±.10**	**84.44±.06**	**84.90±.80**	69.33±.05	68.27±.22	67.25±.39
9	**48.01±.06**	55.34±.17	55.29±.45	73.78±.30	82.77±.07	84.13±.28	71.92±.33	70.66±.26	69.08±.80
12	47.87±.06	**55.61±.04**	55.98±.13	72.29±.25	81.59±.20	83.59±.62	73.33±.16	72.00±.20	70.18±.67
15	47.15±.13	55.49±.15	55.96±.09	70.98±.24	80.69±.08	82.81±.19	73.79±.27	72.87±.03	70.87±.23
18	47.02±.13	55.43±.12	**56.43±.17**	70.13±.06	79.97±.12	82.21±.21	74.63±.11	73.77±.13	72.04±.30
21	46.26±.19	55.31±.20	56.07±.21	68.95±.38	79.25±.23	81.74±.12	**75.17±.28**	**74.15±.38**	**72.11±.12**
Adversarial Training [MMS+18]									
1.00	47.99±.16	50.87±.42	50.12±.13	**77.30±.01**	**85.82±.01**	**85.62±.81**	66.48±.24	62.23±.42	61.62±.46
1.25	**49.24±.12**	53.10±.09	51.97±.46	74.04±.47	84.73±.22	86.25±.12	70.34±.54	65.24±.08	62.94±.35
1.50	49.11±.03	54.15±.03	53.25±.52	72.16±.25	84.35±.19	85.50±.57	72.10±.11	66.65±.06	64.51±.72
1.75	48.32±.63	**54.36±.14**	53.65±.80	70.66±.46	83.95±.30	85.52±.24	72.43±.40	67.31±.03	65.67±.10
2.00	47.44±.06	54.10±.15	**55.78±.22**	69.67±.09	83.49±.06	85.41±.13	**72.73±.04**	**67.53±.01**	**65.71±.15**

as shown by the bottom part of Table 5.1. Proper adaptations should be made to boost the robust regularization for original (generalized) adversarial training. We show the detail of the adaptations in the Appendix. As shown by the table, the large improvements on both TRADES and adversarial training using our boosting strategy suggest that adopting larger λ is crucial in unleashing the full potential of wide models, which is usually neglected in practice.

Table 5.2: Robust accuracy (%) for different datasets, architectures and regularization parameters under various attacks. The highest results are evaluated for three times of randomly started attack. Our approach of boosting regularization for wider models apply to all cases. The value of w and k represents the network width.

Dataset	Architecture	widen-factor/ growth-rate	regularization	PGD [MMS+18]	C&W [CH20a]	FAB [CH20a]	Square [ACFH19]
CIFAR10	WideResNet-34	$w = 1$	$\lambda = 6$	**47.92±.01**	**44.95±.03**	**44.31±.04**	**49.25±.02**
			$\lambda = 12$	47.91±.04	44.24±.02	43.71±.05	47.75±.02
			$\lambda = 18$	46.92±.05	43.48±.03	43.00±.01	46.01±.05
		$w = 5$	$\lambda = 6$	54.50±.03	53.14±.03	52.13±.05	56.79±.02
			$\lambda = 12$	**55.56±.04**	**53.28±.04**	**52.55±.02**	**56.88±.05**
			$\lambda = 18$	55.21±.02	52.64±.02	52.18±.01	56.31±.01
		$w = 10$	$\lambda = 6$	54.23±.04	54.02±.03	52.68±.07	57.64±.03
			$\lambda = 12$	55.80±.06	54.41±.01	53.57±.04	57.72±.10
			$\lambda = 18$	**56.29±.10**	**54.57±.02**	**54.06±.02**	**58.04±.05**
	DenseNet-BC-40	$k = 12$	$\lambda = 6$	**44.79±.02**	40.83±.03	**40.07±.03**	**45.66±.05**
			$\lambda = 12$	44.66±.03	**40.91±.03**	39.88±.01	44.23±.04
			$\lambda = 18$	44.38±.05	40.63±.03	39.42±.01	43.31±.04
		$k = 64$	$\lambda = 6$	55.51±.01	52.76±.04	51.74±.02	57.24±.01
			$\lambda = 12$	**55.85±.03**	**52.98±.02**	**52.10±.03**	**57.34±.04**
			$\lambda = 18$	55.71±.03	52.83±.06	51.66±.04	55.21±.03
CIFAR100	WideResNet-34	$w = 1$	$\lambda = 6$	**24.28±.02**	**20.24±.01**	**19.97±.02**	**22.91±.02**
			$\lambda = 12$	24.18±.04	20.15±.02	19.83±.01	22.78±.01
			$\lambda = 18$	23.99±.03	20.01±.02	19.01±.01	22.04±.01
		$w = 5$	$\lambda = 6$	30.73±.03	27.25±.05	26.01±.03	30.11±.03
			$\lambda = 12$	**31.57±.02**	**27.83±.02**	**27.08±.01**	**30.45±.01**
			$\lambda = 18$	31.38±.01	27.66±.04	26.94±.03	30.02±.01
		$w = 10$	$\lambda = 6$	30.48±.02	27.98±.01	27.00±.11	30.45±.06
			$\lambda = 12$	31.75±.09	29.25±.04	28.14±.03	31.23±.04
			$\lambda = 18$	**32.98±.03**	**29.83±.01**	**28.78±.02**	**32.02±.01**

5.5.3 Experiments on Different Datasets and Architectures

To show that our theory is universal and is applicable to various datasets and architectures, we conduct extra experiments on the CIFAR100 dataset and DenseNet model [HLvdMW17]. For the DenseNet models, the growth rate k denotes how fast the number of channels grows and thus becomes a suitable measure of network width. Following the original paper [HLvdMW17], we choose DenseNet-BC-40 and use models with different growth rates to verify our theory.

Experimental results are shown in Table 5.2. For completeness, we also report the results under four different attack methods and settings, including PGD [MMS+18], C&W [CW17], FAB [CH20a], and Square [ACFH19]. We adopt the best λ from Table 5.1 and show the corresponding performance on models with different widths. It can be seen that our strategy of using a larger robust regularization parameter works very well across different datasets and networks. On the WideResNet model, we observe clear patterns as in Section 5.5.2. On the DenseNet model, although the best regularization λ is different from that of WideResNet, wider models, in general, still require larger λ for better robustness. On CIFAR100, our strategy raises the standard PGD score of the widest model from 30.48% to 32.98%.

5.5.4 Width Adjusted Regularization

Algorithm 8 Width Adjusted Regularization

```
1: Input: initial weights $\boldsymbol{\theta}_0$, WAR parameter $\zeta$, learning rate $\eta$, adversarial attack $\mathcal{A}$
2: $\lambda_0 = 0$, $\alpha = 0.1$
3: for $t = 1, \ldots, T$ do
4:     Get mini-batch $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
5:     for $i = 1, \ldots, m$ (in parallel) do
6:         $\widehat{\mathbf{x}}_i \leftarrow \mathcal{A}(\mathbf{x}_i)$
7:         $l_{\text{nat}} \leftarrow \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{x}_i, y_i)$
8:         $l_{\text{rob}} \leftarrow \mathcal{L}(\boldsymbol{\theta}_t; \widehat{\mathbf{x}}_i, y_i) - \mathcal{L}(\boldsymbol{\theta}_t; \mathbf{x}_i, y_i)$
9:         $\lambda_t \leftarrow \max(\lambda_{t-1} + \alpha \cdot (\zeta - (l_{\text{nat}}/l_{\text{rob}}), 0)$
10:        $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - (\eta/m) \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} [l_{\text{nat}} + \lambda_t \cdot l_{\text{rob}}]$
11:    end for
12: end for
```

Our previous analysis has shown that larger model width may hurt adversarial robustness without properly choosing the regularization parameter λ. However, exhaustively cross-validating λ on wider networks can be extremely time-consuming in practice. To address this issue, we investigate the possibility of automatically adjusting λ according to the model width, based on our existing knowledge obtained in fine-tuning smaller networks, which is much cheaper. Note that the key to achieving the best robustness is to well balance between the natural risk term and the robust regularization term in (5.1.1). Although the regularization parameter λ cannot be directly applied from thinner networks to wider networks (as suggested by our analyses), the best ratio between the natural risk and the robust regularization across different width models can be kept roughly the same. Following this idea, we design the **W**idth **A**djusted **R**egularization (WAR) method, which is summarized in Algorithm 8. Specifically, we first manually tune the best λ for a thin network and record the ratio ζ between the natural risk and the robust regularization when the training converges.

Then, on training wider networks, we adaptively[1] adjust λ to encourage the ratio between the natural risk and the robust regularization to stay close to ζ. Let's take an example here. We first cross-validate λ on a thin network with widen factor 0.5 and identify the best $\lambda = 6$ and $\zeta = 30$ with 18 GPU hours in total. Now we compare three different strategies for training wider models and summarize the results in Table 5.3: 1) directly apply $\lambda = 6$ with no fine-tuning on the current model; 2) exhaustive manual fine-tuning from $\lambda = 6.0$ to $\lambda = 21.0$ (6 trials) as in Table 5.1; 3) our WAR strategy. Table 5.3 shows that the final λ generated by WAR on wider models are consistent with the exhaustively tuned best λ. Compared to the exhaustive manual tuning strategy, WAR achieves even slightly better model robustness with much less overall training time ($\sim$4 times speedup for WRN-34-10 model). On the other hand, directly using $\lambda = 6$ with no tuning on the wide models leads to much worse model robustness while having the same overall training time. This verifies the effectiveness of our proposed WAR method.

5.5.5 Comparison of Robustness on Wide Models

Previous experiments in Section 5.5.2 and Section 5.5.3 have shown the effectiveness of our proposed strategy on using larger robust regularization parameter for wider models. In order to ensure that this strategy does not lead to any obfuscated gradient problem [ACW18] and gives a false sense of robustness, we further conduct experiments using stronger attacks. In particular, we choose to evaluate our best models on the AutoAttack algorithm [CH20b], which is an ensemble attack method that contains four different white-box and black-box attacks for the best attack performances.

We evaluate models trained with WAR, with or without extra unlabeled data [CRS$^+$19], and report the robust accuracy in Table 5.4. Note that the results of other baselines are

[1]the learning rate α for λ_t in Algorithm 8 is not sensitive and needs no extra tuning.

Table 5.3: Comparison of TRADES with different tuning strategies. N/A denotes no fine–tuning of the current model (tuning on small networks only). Manual represents exhaustive fine-tuning.

Model	Tuning	λ	PGD	GPU hours
WRN-34-1	N/A	6.00	47.81	12+18=30
	Manual	9.00	48.01	12×6=72
	WAR	9.12	**48.06**	12+18=30
WRN-34-5	N/A	6.00	54.45	20+18=38
	Manual	12.00	55.61	20×6=120
	WAR	14.37	**55.62**	20+18=38
WRN-34-10	N/A	6.00	54.18	32+18=50
	Manual	18.00	56.43	32×6=192
	WAR	16.43	**56.46**	32+18=50

directly obtained from the AutoAttack leaderboard[2]. From Table 5.4, we can see that our WAR significantly improves the baseline TRADES models on WideResNet. This experiment further verifies the effectiveness of our proposed strategy.

5.6 Proof of Lemma 5.4.2

Lemma 5.6.1 (Restatement of Lemma 5.4.2)**.** For any given input $\mathbf{x} \in \mathbb{R}^d$ and ℓ_2 norm perturbation limit ϵ, if $m \geq \max(d, \Omega(H\log(H)))$, $\frac{R}{\sqrt{m}}+\epsilon \leq \frac{c}{H^6(\log m)^3}$ for some sufficient small c, then with probability at least $1 - O(H)e^{-\Omega(m(R/\sqrt{m}+\epsilon)^{2/3}H)}$, we have for any $\mathbf{x}' \in \mathbb{B}(\mathbf{x}, \epsilon)$ and Lipschitz loss $\mathcal{L}$, the input gradient norm satisfies

$$\|\nabla\mathcal{L}(f(\mathbf{x}'), y)\|_2 = O\big(\sqrt{mH}\big).$$

Proof. The major part of this proof is inspired from [GCL+19]. Let $\mathbf{D}^{(h)}(\mathbf{W}, \mathbf{x}) =$

[2]`https://github.com/fra31/auto-attack`

Table 5.4: Robust accuracy (%) comparison on CIFAR10 under AutoAttack. † indicates training with extra unlabeled data.

Methods	Model	AutoAttack
TRADES [ZYJ+19]	WRN-34-10	53.08
Early-Stop [RWK20b]	WRN-34-20	53.42
FAT [ZXH+20b]	WRN-34-10	53.51
HE [PYD+20]	WRN-34-20	53.74
WAR	WRN-34-10	**54.73**
MART [WZY+20]†	WRN-28-10	56.29
HYDRA [SWMJ20]†	WRN-28-10	57.14
RST [CRS+19]†	WRN-28-10	59.53
WAR†	WRN-28-10	**60.02**
WAR†	WRN-28-20	**61.84**

$\mathrm{diag}(\mathbb{1}\{\mathbf{W}^{(h)}\sigma(\cdots\sigma(\mathbf{W}^{(1)}\mathbf{x})) > 0\})$ be a diagonal sign matrix. Then the neural network function can be rewritten as follows:

$$f(\mathbf{x}) = \mathbf{a}^\top \mathbf{D}^{(H)}(\mathbf{W}, \mathbf{x})\mathbf{W}^{(H)} \cdots \mathbf{D}^{(1)}(\mathbf{W}, \mathbf{x})\mathbf{W}^{(1)}\mathbf{x}.$$

By the chain rule of the derivatives, the input gradient norm can be further written as

$$\begin{aligned}
\|\nabla \mathcal{L}(f(\mathbf{x}'), y)\|_2 &= \|\mathcal{L}'(f(\mathbf{x}'), y) \cdot \nabla f(\mathbf{x}')\|_2 \\
&\leq \|\mathcal{L}'(f(\mathbf{x}'), y)\|_2 \cdot \|\nabla f(\mathbf{x}')\|_2 \\
&= \|\mathcal{L}'(f(\mathbf{x}'), y)\|_2 \cdot \|\mathbf{a}^\top \mathbf{D}^{(H)}(\mathbf{W}, \mathbf{x}')\mathbf{W}^{(H)} \cdots \mathbf{D}^{(1)}(\mathbf{W}, \mathbf{x}')\mathbf{W}^{(1)}\|_2. \qquad (5.6.1)
\end{aligned}$$

Now let us focus on the term $\|\mathbf{a}^\top \mathbf{D}^{(H)}(\mathbf{W}, \mathbf{x}')\mathbf{W}^{(H)} \cdots \mathbf{D}^{(1)}(\mathbf{W}, \mathbf{x}')\mathbf{W}^{(1)}\|_2$. Note that by triangle inequality,

$$
\begin{aligned}
&\|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(1)}\|_2 \\
&\le \|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(1)} - \mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(1)}\|_2 \\
&\qquad + \|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(1)}\|_2. \qquad (5.6.2)
\end{aligned}
$$

Note that $\mathbf{W}$ is updated via projected gradient descent with projection set $B(R)$. Therefore, by Equation (12) in Lemma A.5 of [GCL+19] we have

$$
\begin{aligned}
&\|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(1)} - \mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(1)}\|_2 \\
&= O\bigg((\frac{R}{\sqrt{m}}+\epsilon)^{1/3} H^2\sqrt{m\log m}\bigg), \qquad (5.6.3)
\end{aligned}
$$

and by Lemma A.3 in [GCL+19] we have

$$
\|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W}_0,\mathbf{x})\mathbf{W}_0^{(1)}\|_2 = O(\sqrt{m}H). \qquad (5.6.4)
$$

Combining (5.6.2), (5.6.3), (5.6.4), when $\frac{R}{\sqrt{m}}+\epsilon \le \frac{c}{H^6(\log m)^3}$, we have

$$
\|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(1)}\|_2 = O(\sqrt{m}H). \qquad (5.6.5)
$$

By substituting (5.6.5) into (5.6.1) we have,

$$
\|\nabla\mathcal{L}(f(\mathbf{x}'),y)\|_2 \le \|\mathcal{L}'(f(\mathbf{x}'),y)\|_2 \cdot \|\mathbf{a}^\top\mathbf{D}^{(H)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(H)}\cdots\mathbf{D}^{(1)}(\mathbf{W},\mathbf{x}')\mathbf{W}^{(1)}\|_2 = O(\sqrt{m}H),
$$

where the last inequality holds since $\|\mathcal{L}'(f(\mathbf{x}'),y)\|_2 = O(1)$ due to the Lipschitz condition of loss $\mathcal{L}$. This concludes the proof. $\square$

5.7 The Experimental Detail for Reproducibility

All experiments are conducted on a single NVIDIA V100 GPU. It runs on the GNU Linux Debian 4.9 operating system. The experiment is implemented via PyTorch 1.6.0. We adopt the public released codes of PGD [MMS+18], TRADES [ZYJ+19], and RST [CRS+19] and adapt them for our own settings, including inspecting the loss value of robust regularization and the local Lipschitzness.

CIFAR100 contains 50k images for 100 classes, which means that it has much fewer images for each class compared with CIFAR10. This makes the learning problem of CIFAR100 much harder. For DenseNet architecture, we adopt the 40 layers model with the bottleneck design, which is the DenseNet-BC-40. It has three building blocks, with each one having the same number of layers. This is the same architecture tested in the original paper of DenseNet for CIFAR10. For simplicity reason, we make the training schedule stay the same with the one used for WideResNet, which is the decay learning rate schedule. As DenseNet gets deeper, its channel number (width) will be multiplied with the growing rate k. Thus, as k gets larger, the width of DenseNet also does. Although this mechanism slightly differs from the widen factor of WideResNet, which amplify all layers with the same ratio.

5.8 The Exponential Decay Learning Rate

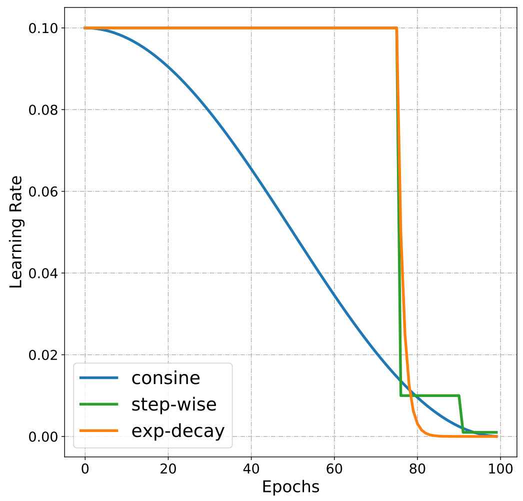


Figure 5.5: The changing trend leanring rate against training epochs for different learning rate schedule.

To demonstrate the fact that the over-fitting problem all comes from perturbation stability in Section 3.2(3), we use the training schedule of the original work for Figure 2. Aside from that, all the other experiments and plots are results under our proposed learning rate schedule, which halve the learning rate for every epochs after the 75-th epoch and can prevent over-fitting. Different learning rate schedules are shown in Figure 5, including the step-wise [ZYJ+19], cosine [CRS+19], and our exp-decay learning rate schedule. Basically, our schedule is an early-stop version of the baseline of TRADES [ZYJ+19], which skips the small learning rate stage as soon as possible in the later stage. We found this schedule is the most effective one when only training on the original CIFAR10. However, when combined with the 500K unlabeled images from RST [CRS+19], we find that the over-fitting problem is much less severe and cosine learning rate is the best choice.

5.9 Boosting the Original Adversarial Training

We further show that our strategy also applies to the original adversarial training [MMS+18]. Note that our generalized adversarial training framework (5.1.1) allow us to further boost the robust regularization for original (generalized) adversarial training. The only caveat is that in adversarial training formulation, the robust regularization term is not guaranteed to be non-negative in practice[3]. To avoid this problem, we manually set the robust regularization term in (5.1.1) to be non-negative by clipping the $\mathcal{L}(\boldsymbol{\theta}; \widehat{\mathbf{x}}, y) - \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y)$ term. Let us denote $\mathbf{x}'$ as the empirical maximization solution, the final loss function becomes:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \Big\{ \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y) + \lambda \cdot \max_{\widehat{\mathbf{x}}_i \in \mathbb{B}(\mathbf{x}_i, \epsilon)} \big(\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}', y) - \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y), 0 \big) \Big\}.$$

The bottom part of Table 5.1 shows the experimental results for boosting the robust regularization parameter for (generalized) adversarial training models. We can observe that the boosting strategy still works in this method, and wider models indeed require larger λ

[3]Successfully solving the inner maximization problem in (5.1.1) is supposed to guarantee that $\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}', y) > \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y)$, however, in practice, there still exist a very little chance that $\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}', y) < \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}, y)$ due to failure in solving the inner maximization problem at the beginning of the training procedure with limited steps.

to obtain the best robust accuracy.

5.10 Conclusions

In this paper, we studied the relation between network width and adversarial robustness in adversarial training, a principled approach to train robust neural networks. We showed that the model robustness is closely related to both natural accuracy and perturbation stability, while the balance between the two is controlled by the robust regularization parameter λ. With the same value of λ, the natural accuracy is better on wider models while the perturbation stability actually becomes worse, leading to a possible decrease in the overall model robustness. We showed the origin of this problem by relating perturbation stability with local Lipschitzness and leveraging recent studies on the neural tangent kernel to prove that larger network width leads to worse perturbation stability. Our analyses suggest that: 1) proper tuning of λ on wider models is necessary despite being extremely time-consuming; 2) practitioners should adopt a larger λ for training wider networks. Finally, we propose the Width Adjusted Regularization, which significantly saves the tuning time for robust training on wide models.

CHAPTER 6

Backward Smoothing for Efficient Robust Training

6.1 Introduction

Deep neural networks are well known to be vulnerable to adversarial examples [SZS$^+$13], *i.e.*, a small perturbation on the original input can lead to misclassification or erroneous prediction. Many defense methods have been developed to mitigate the disturbance of adversarial examples [GRCVDM18, XWZ$^+$18, SKN$^+$18, MLW$^+$18, SKC18, DAL$^+$18, MMS$^+$18, ZYJ$^+$19], among which robust training methods, such as adversarial training [MMS$^+$18] and TRADES [ZYJ$^+$19], are currently the most effective strategies. Specifically, adversarial training method [MMS$^+$18] trains a model on adversarial examples by solving a min-max optimization problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{x}_i' \in \mathcal{B}_\epsilon(\mathbf{x}_i)} L(f_{\boldsymbol{\theta}}(\mathbf{x}_i'), y_i), \tag{6.1.1}$$

where $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is the training dataset, $f(\cdot)$ denotes the logits output of the neural network, $\mathcal{B}_\epsilon(\mathbf{x}_i) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_i\|_\infty \le \epsilon\}$ denotes the ϵ-perturbation ball, and L is the cross-entropy loss.

On the other hand, instead of directly training on adversarial examples, TRADES [ZYJ$^+$19] further improves model robustness with a trade-off between natural accuracy and robust accuracy, by solving the empirical risk minimization problem with a robust regularization term:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \Big[L(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \beta \max_{\mathbf{x}_i' \in \mathcal{B}_\epsilon(\mathbf{x}_i)} \mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x}_i)), s(f_{\boldsymbol{\theta}}(\mathbf{x}_i'))\big) \Big], \tag{6.1.2}$$

where $s(\cdot)$ denotes the softmax function, and $\beta > 0$ is a regularization parameter. The goal of this robust regularization term (*i.e.*, KL divergence term) is to ensure the outputs are stable within the local neighborhood. Both adversarial training and TRADES achieve good model robustness, as shown on recent model robustness leaderboards[1] [CH20b, CG20]. However, a major drawback lies in that both are highly time-consuming for training, limiting their usefulness in practice. This is largely due to the fact that both methods perform iterative adversarial attacks (*i.e.*, Projected Gradient Descent) to solve the inner maximization problem in each outer minimization step.

Recently, [WRK20] shows that it is possible to use single-step adversarial attacks to solve the inner maximization problem, which previously was believed impossible. The key ingredient in their Fast AT approach is adding a random initialization step before the single-step adversarial attack. This simple change leads to a reasonably robust model that outperforms other fast robust training techniques, *e.g.*, [SNG$^+$19]. However, the simple change also has its downsides: 1) random initialization makes single-step robust training possible yet it can be quite unstable [LWJC20]; 2) compared to state-of-the-art robust training models [MMS$^+$18, ZYJ$^+$19], Fast AT still lags behind on model robustness. Besides these, It also remains a mystery in [WRK20] on why random initialization is empirically effective.

Although some attempts have been made trying to explain the role of random initialization and further improve Fast AT [AF20, LWJC20], in this work, we aim to understand the role of random initialization in [WRK20] from a new perspective and further close the robustness gap between standard adversarial training and Fast Adversarial Training [WRK20]. Specifically, We propose a new principle towards understanding Fast AT - that random initialization can be viewed as performing randomized smoothing for better optimization of the inner maximization problem. In order to further improve the model robustness of fast robust training techniques, we propose a new initialization strategy, *backward smoothing*, which strengthens the smoothing effect within the ϵ-perturbation ball. The resulting method

[1]`https://github.com/fra31/auto-attack` and `https://github.com/uclaml/RayS`.

significantly improves both stability and model robustness over the single-step random initialization strategies. Moreover, even comparing with full-step robust training methods such as TRADES [ZYJ$^+$19], our proposed backward smoothing strategy achieves similar model robustness while consuming much less training time ($\sim$ 3x improvement with the same training schedule).

The remainder of this paper is organized as follows: in Section 6.2, we briefly review existing literature on adversarial attacks, robust training as well as randomized smoothing technique. We present our new understanding of random initialization in Section 6.3. We present our proposed method in Section 6.4. In Section 6.5, we empirically evaluate our proposed method with other state-of-the-art baselines. Finally, we conclude this paper in Section 6.8.

6.2 Related Work

There exists a large body of work on adversarial attacks and defenses. In this section, we only review the most relevant work to ours.

Adversarial Attack The concept of adversarial examples was first proposed in [SZS$^+$13]. Since then, many methods have been proposed, such as Fast Gradient Sign Method (FGSM) [GSS15], and Projected Gradient Descent (PGD) [KGB16, MMS$^+$18]. Later on, various attacks [PMJ$^+$16, MDFF16, CW17, ACW18, CZYG20, CH20a, SABB20, TSE20] were also proposed for better effectiveness or efficiency.

There are also many attacks focused on different attack settings. [CZS$^+$17] proposed a black-box attack where the gradient is not available, by estimating the gradient via finite-differences. Various methods [IEA$^+$18, ADO20, MAS19, ACFH19, TSE20] have been developed to improve the query efficiency of [CZS$^+$17]. Other methods [BRB18, CLC$^+$19, CSC$^+$20] focused on the more challenging hard-label attack setting, where only the prediction labels are available. On the other hand, there is recent work [CH20b, CG20] that aims to accurately evaluate the model robustness via an ensemble of attacks or effective hard-label attack.

Robust Training Many heuristic defenses [GRCVDM18, XWZ$^+$18, SKN$^+$18, MLW$^+$18, SKC18, DAL$^+$18] were proposed when the concept of adversarial examples was first introduced. However, they are later shown by [ACW18] as not truly robust. Adversarial training [MMS$^+$18] is the first effective method towards defending against adversarial examples. Various adversarial training variants [WMB$^+$19, WZY$^+$20, ZYJ$^+$19, WXW20, SABB20, ZXH$^+$20a] were later proposed to further improve the adversarially trained model robustness. A line of researches focus on studying various others factors affecting model robustness such as early-stopping [RWK20a], model width [WCC$^+$20], loss landscape [LSL$^+$20] and parameter tuning [PYD$^+$21, GQU$^+$20]. Another line of research utilizes extra information (*e.g.*, pre-trained models [HLM19] or extra unlabeled data [CRS$^+$19, AUH$^+$19]) to further improve robustness.

Recently, many focus on improving the training efficiency of adversarial training based algorithms, such as free adversarial training [SNG$^+$19] and Fast AT [WRK20], which uses single-step attack (FGSM) with random initialization. [LWJC20] proposed a hybrid approach for improving Fast AT which is orthogonal to ours. [AF20] proposed a new regularizer promoting gradient alignment for more stable training. Yet, it is not focused on closing the robustness gap with state-of-the-arts.

Randomized Smoothing [DBW12] proposed the randomized smoothing technique and proved variance-based convergence rates for non-smooth optimization. Later on, this technique was applied to certified adversarial defenses [CRK19, SLR$^+$19] for building robust models with certified robustness guarantees. In this paper, we are not targeting certified defenses. Instead, we use the randomized smoothing concept in optimization to explain Fast AT.

6.3 Pros and Cons of Random Initialization

In this section, we analyze the pros and cons of random initialization in Fast AT [WRK20]. First, let us explain why random initialization in Fast AT is effective by looking into why one-step AT would fail without random initialization.

6.3.1 What Caused the Failure of One-step AT Without Random Initialization?

[WRK20] has already shown that without random initialization, one-step AT would almost surely fail in the training procedure due to catastrophic overfitting, i.e., the robust accuracy w.r.t. a PGD adversarial suddenly drops to near 0 even on training data. However, it is not clear what exactly cause this phenomenon. One natural conjecture is that perhaps the one-step attack is not effective enough for adversarial training purposes. Recall that the perturbation is obtained by solving the following inner maximization problem in adversarial training:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta} \in \mathcal{B}_\epsilon(\mathbf{0})}{\operatorname{argmax}} L(f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\delta}), y). \tag{6.3.1}$$

To figure out whether the attack effectiveness is the key cause for the poor performance of the plain one-step AT without random initialization, we conduct the following simple experiments by observing the loss increment after attack in each training step, i.e.,

$$\Delta_L = L(f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\delta}^*), y) - L(f_{\boldsymbol{\theta}}(\mathbf{x}), y),$$

where $\{(\mathbf{x}, y)\}$ is the clean training example and $\boldsymbol{\delta}^*$ is the solution from (6.3.1). Since (6.3.1) aims at maximizing the loss value, this loss increment term Δ_L should always be positive along the entire training trajectory.

In Figure 6.1, we plot the loss increment Δ_L for three different training trajectories: Fast AT without random initialization, Fast AT with random initialization, as well as standard AT. We observe that with the random initialization, Fast AT's loss increment is quite close to standard AT (although it still can go wrong from time to time). However, without random

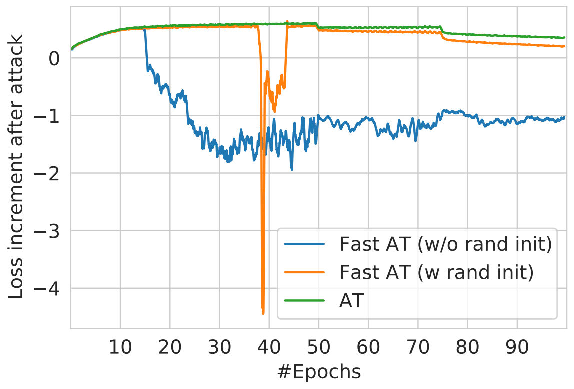


Figure 6.1: Loss increment after attack, i.e., $L(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}^*), y) - L(f_{\boldsymbol{\theta}}(\mathbf{x}), y)$, along the training trajectory for different methods on training ResNet-18 on CIFAR-10 dataset.

initialization, the one-step attack is more than just not effective: the loss value after the attack (the inner maximization step) is actually worse than that of clean data. Since Fast AT performs a one-step gradient ascent to solve (6.3.1), this suggests that the step size used in Fast AT for solving (6.3.1) is actually too large. On the other hand, in order to effectively defend against perturbations of magnitude ϵ with only one step attack budget, the attack step size has to be chosen close to ϵ. This is actually quite intuitive: with a small attack step size and only one step attack budget, the generated adversarial examples during the training phase will never reach the magnitude of ϵ. Therefore, when facing perturbations of the magnitude of ϵ during the testing phase, the model stands little chance defending against them. This dilemma explains the cause of failure for one-step AT without random initialization.

6.3.2 Why Random Initialization Helps?

Now let us talk about random initialization. It is well known from optimization theory [BV04] that, for gradient descent-based algorithms, the maximum allowed step size (in order to guarantee convergence) is directly related to the smoothness of the optimization objective function. Specifically, the smoother the objective function is, the larger the gradient step size is allowed. Here we argue that random initialization works just as the randomized smoothing technique [DBW12], which makes the overall optimization objective more smooth via random perturbations of the optimization variable[2].

To see why random initialization works as randomized smoothing here, let us apply randomized smoothing to (6.3.1):

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}+\epsilon\boldsymbol{\xi}\in\mathcal{B}_\epsilon(\mathbf{0})}{\operatorname{argmax}} \mathbb{E}_{\boldsymbol{\xi}\sim U(-1,1)} L(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}+\epsilon\boldsymbol{\xi}), y), \tag{6.3.2}$$

where $\boldsymbol{\xi}$ is the perturbation vector for randomized smoothing, and $\boldsymbol{\delta}$ is the adversarial perturbation vector (initialized as zero). Suppose we solve (6.3.2) in a stochastic fashion (*i.e.*, sample a random perturbation $\boldsymbol{\xi}$ instead of computing the expectation over $\boldsymbol{\xi}$), and using only one step gradient update. We can see that this reduces to the Fast AT formulation. This suggests that Fast AT can be viewed as performing stochastic single-step attacks on a randomized smoothed objective function which allows the use of a larger step size. This explains why random initialization helps Fast AT as it makes the loss objective smoother and thus easier to optimize with large step sizes such as ϵ.

It is worth noting that [AF20] also provided an explanation of random initialization: it reduces the magnitude of the perturbation and thus the network becomes more linear and fits better toward single-step attack. In fact, our argument is more general and can cover theirs, because if the loss function is approximately linear, then it will be very smooth, *i.e.*, the second-order term in the Taylor expansion is small. And their observations that Fast AT using smaller attack step size can succeed without random initialization actually also validate our analysis above.

[2]Instead of using only the gradient at the original iterate, randomized smoothing proposes to randomly generate perturbed iterates and use their gradients for the optimization procedure. More details about the randomized smoothing technique are provided in Appendix 6.6.

Table 6.1: Model robustness comparison among AT, Fast AT, TRADES and Fast TRADES, using ResNet-18 model on CIFAR-10 dataset.

Method	Nat (%)	Rob (%)
AT	82.36	51.14
Fast AT	84.79	46.30
TRADES	82.33	52.74
Fast TRADES	83.39	46.98

6.3.3 Drawbacks of Random Initialization

Although the random initialization effectively helps Fast AT avoid the catastrophic overfitting from happening in the most time, it still exposes several major weaknesses.

Performance Stability Fast AT can still be highly unstable (*i.e.*, catastrophic overfitting can still occur from time to time). This is also observed in [LWJC20]. In Figure 6.1, we also observe that Fast AT could still fail in solving the inner maximization problem (especially when using a drastically large attack step size). It can be imagined that with some bad luck, the training procedure of Fast AT could still fall apart even with random initialization.

Unfortunately, Fast AT performs better with a larger attack step size. We run Fast AT on CIFAR-10 using ResNet-18 model [HZRS16a] for 10 times[3]. For the best attack step size of 10/255 (according to [WRK20]), the best run achieves 46.30% robust accuracy, however, the average is only 42.11% since many runs actually failed.

Potential for Further Robustness Improvement Fast AT uses standard adversarial training [MMS+18] as the baseline, and can obtain similar robustness performance. However, later work [RWK20a] shows that adversarial training can cause the overfitting problem, while

[3]Here we exclude the additional acceleration techniques in [WRK20] and apply standard piecewise learning rate decay as in [MMS+18, ZYJ+19].

early stopping can largely improve robustness. [ZYJ+19] further achieves even better model robustness that is much higher than what Fast AT obtains. From Table 6.1, we observe that there exists a 6% robust accuracy gap between Fast AT and the best-performing TRADES model even for the best run over 10 repeats. This indicates that Fast AT is still far from optimal, and there is still big room for further robustness improvement.

6.4 Proposed Approaches

6.4.1 A Naive Try: Randomized Smoothing for TRADES

In the previous section, we show that objective smoothness plays a key role in the success of single-step adversarial training. Note the TRADES [ZYJ+19] method naturally promotes the objective smoothness in its training formula (by minimizing the output discrepancy of input examples within the perturbation ball). From this perspective, it should be more fit to single-step robust training than AT. What's more, as shown in Table 6.1, TRADES enjoys better model robustness compared with standard AT. Therefore we first try to apply randomized smoothing to TRADES and see if this leads to better robust training method. Let us recall the inner maximization formulation for TRADES:

$$\max_{\boldsymbol{\delta}\in\mathcal{B}_\epsilon(\mathbf{0})} \mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x})), s(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}))\big). \tag{6.4.1}$$

Similarly, we can smooth this objective and solve the following objective instead:

$$\max_{\boldsymbol{\delta}\in\mathcal{B}_\epsilon(\mathbf{0})} \mathbb{E}_{\boldsymbol{\xi}\sim U(-1,1)}\mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x})), s(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}+\epsilon\boldsymbol{\xi}))\big). \tag{6.4.2}$$

This leads to the same adversarial example formulation as using random initialization and then performing single-step projected gradient ascent. We refer to this strategy as Fast TRADES. We experimentally test Fast TRADES by training the ResNet-18 model on the CIFAR-10 dataset. From Table 6.1, we can see that Fast TRADES indeed achieves better performance than Fast AT. Yet the improvement is not significant and still falls far behind

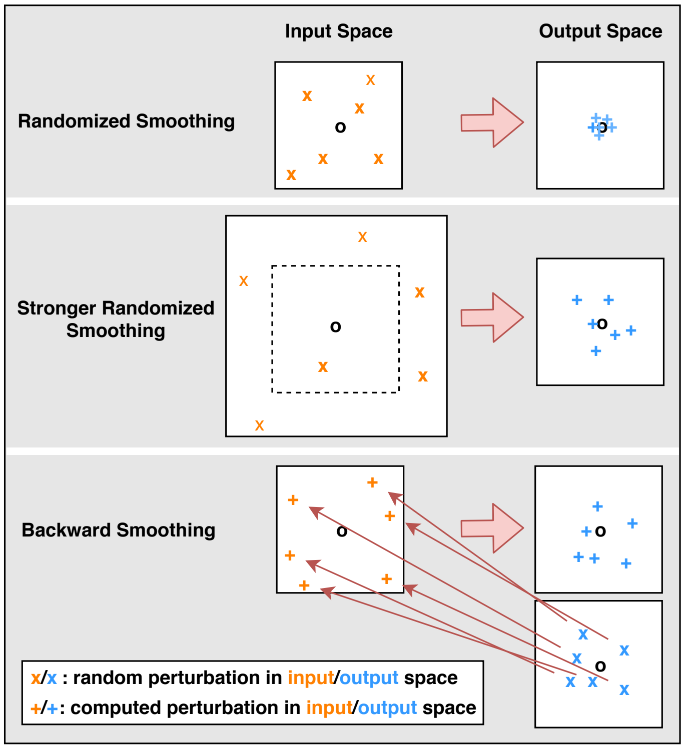

Input Space
Output Space
Randomized Smoothing
Stronger Randomized Smoothing
Backward Smoothing
x/x : random perturbation in input/output space
+/+: computed perturbation in input/output space

Figure 6.2: A sketch of our proposed method.

the original TRADES method. This inspires us to study how to design a better strategy for more significant improvements.

According to our previous analysis in Section 6.3, one way to further improve the robust training performances is to further strengthen the smoothing effect. However, unlike the general randomized smoothing setting, one of the special constraints in the adversarial setting is that random perturbation on the input vector is subject to the ϵ-ball constraint, therefore cannot be too large. This means that we cannot further increase the smoothing effect by simply using larger random perturbations.

6.4.2 Backward Smoothing

Now we introduce our proposed method to address the above issue. The goal is to further boost the smoothing effect of randomized smoothing without violating the ϵ-perturbation constraint. Note that if we are allowed to use larger random perturbations, we expect that $\mathrm{KL}(s(f_{\boldsymbol{\theta}}(\mathbf{x})), s(f_{\boldsymbol{\theta}}(\mathbf{x}+u\boldsymbol{\xi})))$ will also be larger, meaning that the neural network output of the random initialization $f_{\boldsymbol{\theta}}(\mathbf{x}+u\boldsymbol{\xi})$ should be more different from the original output $f_{\boldsymbol{\theta}}(\mathbf{x})$ (as shown in Figure 6.2). This inspires us to generate the initialization point in a backward fashion. Specifically, let us denote the input domain $\mathbf{x} \in \mathbb{R}^d$ as the input space, and their corresponding neural network output $f_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^c$ as the output space, where c is the number of classes for the classifier. We first generate random points in the output space just as randomized smoothing does in the input space, *i.e.*, $f_{\boldsymbol{\theta}}(\mathbf{x}) + \gamma\boldsymbol{\psi}$, where $\boldsymbol{\psi} \sim U(-1, 1)$ is the random variable and γ is a small number. Then we find the corresponding input perturbation in a backward fashion and use it as our initialization. An illustrative sketch of our proposed method is provided in Figure 6.2.

Now we formalize our proposed method in mathematical language. The key step in our proposed method is to find the input perturbation $\boldsymbol{\xi}$ such that:

$$f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\xi}) = f_{\boldsymbol{\theta}}(\mathbf{x}) + \gamma\boldsymbol{\psi}. \tag{6.4.3}$$

In order to find the best $\boldsymbol{\xi}^*$ to satisfy (6.4.3), we turn to solve the following problem:

$$\boldsymbol{\xi}^* = \operatorname*{argmin}_{\boldsymbol{\xi}\in\mathcal{B}_\epsilon(\mathbf{0})} \mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x}) + \gamma\boldsymbol{\psi}), s(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\xi}))\big). \tag{6.4.4}$$

Note that $\boldsymbol{\xi}$ is initialized as a zero vector. For the sake of computational efficiency, we solve (6.4.4) using single-step PGD in practice. Then, similar to [WRK20], we use single-step gradient update for the inner maximization problem:

$$\boldsymbol{\delta}^* = \operatorname*{argmax}_{\boldsymbol{\delta}+\boldsymbol{\xi}^*\in\mathcal{B}_\epsilon(\mathbf{0})} \mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x})), s(f_{\boldsymbol{\theta}}(\mathbf{x}+\boldsymbol{\delta}+\boldsymbol{\xi}^*))\big). \tag{6.4.5}$$

Finally, we update the neural network parameter $\boldsymbol{\theta}$ using stochastic gradients at the adversarial point $\mathbf{x} + \boldsymbol{\xi}^* + \boldsymbol{\delta}^*$. A summary of our proposed algorithm is provided in Algorithm 9[4].

[4][TSE20] proposed an attack which also samples diversified points in the output space. Yet we argue that the formulation and the intuition are quite different. More details are provided in the Appendix.

Algorithm 9 Backward Smoothing

```
1: input: The number of training iterations $T$, number of adversarial perturbation steps $K$, maximum perturbation strength $\epsilon$, training step size $\eta$, adversarial perturbation step size $\alpha$, regularization parameter $\beta > 0$;
2: Random initialize model parameter $\boldsymbol{\theta}_0$
3: for $t = 1, \dots, T$ do
4:     Sample mini-batch $\{\mathbf{x}_i, y_i\}_{i=1}^m$ from training set
5:     Obtain $\boldsymbol{\xi}^*$ by solving (6.4.4)
6:     Obtain $\boldsymbol{\delta}^*$ by solving (6.4.5)
7:     $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta/m \cdot \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \big[L(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \beta \cdot \mathrm{KL}\big(s(f_{\boldsymbol{\theta}}(\mathbf{x}_i)), s(f_{\boldsymbol{\theta}}(\mathbf{x}_i + \boldsymbol{\xi}^* + \boldsymbol{\delta}^*))\big)\big]$
8: end for
```

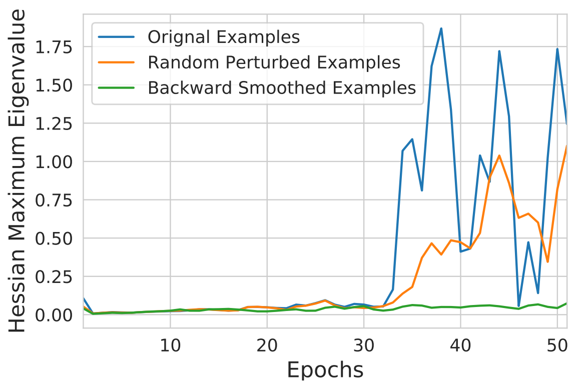


Figure 6.3: Hessian maximum eigenvalue comparison against training epochs.

Figure 6.3 shows the maximum eigenvalue of Hessian of the loss function at the original examples, randomly perturbed examples, and backward smoothed examples along the training trajectory until Fast TRADES obtains its best robustness (the 51st epoch). We observe that during the model training process, the randomly perturbed examples have overall smaller Hessian maximum eigenvalue[5] than that of original examples. This suggests that random smoothing indeed makes the loss function smoother. Moreover, the Hessian maximum eigenvalue under backward smoothing is much smaller than that under random smoothing,

[5]The smaller Hessian maximum eigenvalue, the smoother the loss function is.

showing the insufficiency of the random smoothing techniques and the advantages of our proposed backward smoothing method.

6.5 Experiments

In this section, we empirically evaluate the performance of our proposed method. We first compare our proposed method with other robust training baselines on CIFAR-10, CIFAR100 [KH+09] and Tiny ImageNet [DDS+09][6] datasets. We also provide multiple ablation studies as well as robustness evaluation with state-of-the-art adversarial attack methods to validate that our proposed method provides effective robustness improvement.

6.5.1 Experimental Setting

Following previous work on robust training [MMS+18, ZYJ+19, WRK20], we set $\epsilon = 0.031$ for all three datasets. In terms of model architecture, we adopt standard ResNet-18 model [HZRS16a] for both CIFAR-10 and CIFAR-100 datasets, and ResNet-50 model for Tiny ImageNet. We follow the standard piecewise learning rate decay schedule as used in [MMS+18, ZYJ+19] and set decaying point at 50-th and 75-th epochs. The starting learning rate for all methods is set to 0.1, the same as previous work [MMS+18, ZYJ+19]. For all methods, we tune the models for their best robustness performances (making sure the performance gain is not coming from exploiting the trade-off from natural accuracy). For Adversarial Training and TRADES methods, we adopt 10-step iterative PGD attack with a step size 2/255 for both. For our proposed method, we set the backward smoothing parameter $\gamma = 1$. For robust accuracy evaluation, we typically adopt 100-step PGD attack with the step size 2/255. To ensure the validity of the model robustness improvement is not because of the obfuscated gradient [ACW18], we further test our method with current state-of-the-art attacks

[6]We do not test on ImageNet dataset mainly due to that TRADES does not perform well on ImageNet as mentioned in [QMG+19].

Table 6.2: Performance comparison on CIFAR-10 using ResNet-18 model.

Method	Nat (%)	Rob (%)	Time (m)
AT	82.36	51.14	430
Fast AT	**84.79**	46.30	**82**
Fast AT (2-step)	83.21	49.91	127
Fast AT (GradAlign)	84.37	46.99	402
TRADES	82.33	**52.74**	482
Fast TRADES	83.39	46.98	126
Fast TRADES (2-step)	83.51	48.78	164
Backward Smoothing	82.38	52.50	164

[CH20b, CG20]. All the experiments are conducted on RTX2080Ti GPU servers.

6.5.2 Performance Comparison with Robust Training Baselines

We compare the adversarial robustness of Backward Smoothing against standard Adversarial Training [MMS+18], TRADES [ZYJ+19], as well as fast training methods such as Fast AT [WRK20] and our naive baseline Fast TRADES. We also compare with recently proposed Fast AT+ [LWJC20][7] and GradAlign [AF20][8]. Since our proposed backward smoothing initialization utilizes an extra step of gradient back-propagation, we also compare with Fast AT, Fast TRADES using 2-step attack for fair comparison.

Table 6.2 shows the performance comparison on the CIFAR-10 dataset using ResNet-18 model. Our Backward Smoothing method significantly closes the robustness gap between state-of-the-art robust training methods, achieving high robust accuracy that is almost as

[7]Since [LWJC20] does not have code released yet, we only compare with theirs in the same setting (combined with acceleration techniques) using reported numbers.

[8]We only compare with [AF20] in Tables 6.2, 6.3, 6.7 as its double backpropagation formulation requires much larger memory usage.

Table 6.3: Performance comparison on CIFAR-100 using ResNet-18 model.

Method	Nat (%)	Rob (%)	Time (m)
AT	55.22	28.53	428
Fast AT	**60.35**	24.64	**83**
Fast AT (2-step)	56.00	27.84	128
Fast AT (GradAlign)	58.38	26.26	402
TRADES	56.99	29.41	480
Fast TRADES	60.26	21.33	126
Fast TRADES (2-step)	58.81	25.47	165
Backward Smoothing	56.96	**30.50**	164

Table 6.4: Performance comparison on Tiny ImageNet dataset using ResNet-50 model.

Method	Nat (%)	Rob (%)	Time (m)
AT	44.50	21.34	2666
Fast AT	**49.58**	18.56	**575**
Fast AT (2-step)	45.74	20.94	817
TRADES	47.02	21.04	2928
Fast TRADES	50.36	17.22	805
Fast TRADES (2-step)	46.92	19.26	1045
Backward Smoothing	46.68	**22.32**	1035

good as TRADES, while consuming much less ($\sim$3x) training time. Compared with Fast AT, Backward Smoothing typically costs twice the training time, yet achieving significantly higher model robustness. Notice that the GradAlign method indeed slightly improves upon Fast AT, but it also costs much more training time due to its double backpropagation formulation, making it less competitive to our Backward Smoothing method. Our method also achieves a large performance gain against Fast TRADES. Note that even compared with Fast TRADES using 2-step attack and Fast AT using 2-step attack, which costs about the same training time as ours, our method still achieves a large improvement.

Table 6.3 shows the performance comparison on CIFAR-100 using ResNet-18 model. We can observe patterns similar to the CIFAR-10 experiments. Backward Smoothing achieves slightly higher robustness compared with TRADES, while costing much less training time. Compared with Fast TRADES using 2-step attack and Fast AT using 2-step attack, our method also achieves a large robustness improvement with roughly the same training cost. Table 6.4 shows that on Tiny ImageNet using the ResNet-50 model, Backward Smoothing also achieves significant robustness improvement over other single-step robust training methods.

6.5.3 Evaluation with State-of-the-art Attacks

To ensure that Backward Smoothing does not cause obfuscated gradient problem [ACW18] or presents a false sense of security, we further evaluate our method using state-of-the-art attacks, by considering two evaluation methods: (*i*) AutoAttack [CH20b], which is an ensemble of four diverse (white-box and black-box) attacks (APGD-CE, APGD-DLR, FAB [CH20a] and Square Attack [ACFH19]) to reliably evaluate robustness; (*ii*) RayS attack [CG20], which only requires the prediction labels of the target model (completely gradient-free) and is able to detect falsely robust models. It also measures another robustness metric, average decision boundary distance (ADBD), defined as examples' average distance to their closest decision boundary. ADBD reflects the overall model robustness beyond ϵ constraint. Both evaluations provide online robustness leaderboards for public comparison with other models.

We train our method with WideResNet-34-10 model [ZK16] and evaluate via AutoAttack and RayS. Table 6.5 shows that under state-of-the-art attacks, Backward Smoothing still holds high robustness comparable to TRADES. Specifically, in terms of robust accuracy, Backward Smoothing is only 2% behind TRADES, while significantly higher than AT [MMS$^+$18] and Fast AT [WRK20]. In terms of ADBD metric, Backward Smoothing

Table 6.5: Performance comparison with state-of-the-art robust models on CIFAR-10 evaluated by AutoAttack and RayS.

Method	AutoAttack	RayS	
Metric	Rob (%)	Rob (%)	ADBD
AT (original, no early-stop)	44.04	50.70	0.0344
AT	49.10	54.00	0.0377
Fast AT	43.21	50.10	0.0334
TRADES	**53.08**	**57.30**	**0.0403**
Fast TRADES	43.84	52.05	0.0348
Fast TRADES (2-step)	48.20	54.43	0.0383
Backward Smoothing	51.13	55.08	**0.0403**

achieves the same level of overall model robustness as TRADES, much higher than the other two methods. Note that the gap between Backward Smoothing and TRADES is larger than that in Table 6.2. We want to emphasize that this is not mainly due to the stronger attacks[9] but the fact that we are using larger model architectures. Intuitively speaking, larger models have larger capacities and may need stronger attacks to reach some dark spot in the area.

6.5.4 Stability and Sensitivity

In this subsection, we also study the stability and sensitivity of our proposed Backward Smoothing method.

Training Stability We first take a look into the training stability. In Section 6.3 we have shown that Fast AT can still be highly non-stable in spite of its decent robustness performances. Figure 6.4 shows that Backward Smoothing is much more stable than Fast

[9]We also tested the ResNet-18 models in Table 6.2 with AutoAttack and the gap between Backward Smoothing and TRADES is as small as 0.5%.

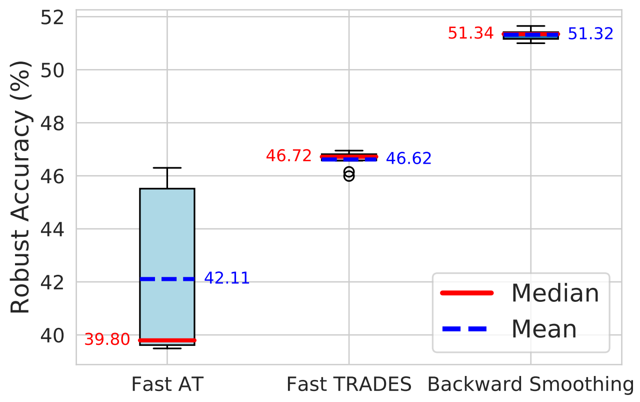


Figure 6.4: Training stability of different fast robust training methods.

AT with much smaller variances. Compared with Fast TRADES, Backward Smoothing has achieved similar variance while obtaining much higher average model robustness. This demonstrates the superiority of our Backward Smoothing method on training stability.

Sensitivity of Attack Step Size We also take a look at the sensitivity of our Backward Smoothing method with various attack step sizes. From Table 6.6, we can observe that unlike Fast AT, which typically enjoys better robustness with larger step size (until it is too large and failed in training), Backward Smoothing achieves similar robustness with a slightly smaller step size, while the best performance is obtained with step size 8/255. This suggests that we do not need to pursue overly-large step size for better robustness as in Fast AT. This also helps avoid the stability issue in Fast AT.

6.5.5 Combining with Other Acceleration Techniques

Aside from random initialization, [WRK20] also adopts two additional acceleration techniques to further improve training efficiency with a minor sacrifice on robustness performance: cyclic learning rate decay schedule [Smi17] and mix-precision training [MNA$^+$17]. We show that such strategies are also applicable to Backward Smoothing. Table 6.7 provides the results when these acceleration techniques are applied. We can observe that both work

Table 6.6: Sensitivity analysis of the attack step size on the CIFAR-10 and CIFAR-100 datasets using ResNet-18 model.

Dataset	CIFAR-10		CIFAR-100	
Step Size	Nat (%)	Rob (%)	Nat (%)	Rob (%)
6/255	81.38	52.38	56.83	29.78
7/255	81.96	52.40	56.61	29.82
8/255	82.38	**52.50**	56.96	**30.50**
9/255	82.47	52.16	56.45	29.35
10/255	81.71	52.04	60.85	24.21
11/255	67.43	42.45	40.40	20.92
12/255	65.56	41.12	37.90	18.83

universally well for all methods, significantly reducing training time (in comparison with Table 6.2). Yet it does not alter the conclusions that Backward Smoothing achieves similar robustness to TRADES with much less training time. Also when compared with the recent proposed Fast AT+ method, Backward Smoothing achieves higher robustness and training efficiency. Note that the idea of the Fast AT+ method is orthogonal to ours and we can also adopt such a hybrid approach for further reduction on training time.

6.6 Randomized Smoothing

Randomized smoothing technique [DBW12] was originally proposed for solving convex non-smooth optimization problems. It is based on the observations that random perturbation of the optimization variable can be used to transform the loss into a smoother one. Instead of using only $L(\mathbf{x})$ and $\nabla L(\mathbf{x})$ to solve

$$\min L(\mathbf{x}),$$

Table 6.7: Performance comparison on CIFAR-10 using ResNet-18 model combined with cyclic learning rate and mix-precision training.

Method	Nat (%)	Rob (%)	Time (m)
AT	81.48	50.32	62
Fast AT	83.26	45.30	**12**
Fast AT+	83.54	48.43	28
Fast AT (GradAlign)	81.80	46.90	54
TRADES	79.64	**50.86**	88
Fast TRADES	**84.40**	45.96	18
Fast TRADES (2-step)	81.37	47.56	24
Backward Smoothing	78.76	50.58	24

randomized smoothing turns to solve the following objective function, which utilizes more global information from neighboring areas:

$$\min \mathbb{E}_{\boldsymbol{\xi}\sim U(-1,1)} L(\mathbf{x} + u\boldsymbol{\xi}), \tag{6.6.1}$$

where $\boldsymbol{\xi}$ is a random variable, and u is a small number. [DBW12] showed that randomized smoothing makes the loss in (6.6.1) smoother than before. Hence, even if the original loss L is non-smooth, it can still be solved by stochastic gradient descent with provable guarantees.

6.7 Additional Ablation Studies

In this section, we conduct additional ablation studies to provide a comprehensive view to the Backward Smoothing method.

6.7.1 Does Backward Smoothing alone works?

To further understand the role of Backward Smoothing in robust training, we conduct experiments on using Backward Smoothing alone, i.e., only use Backward Smoothing initialization but do not perform gradient-based attack at all. Table 6.8 and Table 6.9 show the experimental results. We can observe that Backward Smoothing as an initialization itself only provides a limited level of robustness (not as good as a single-step attack). This is reasonable since the loss for Backward Smoothing does not directly promote adversarial attacks. Therefore it only serves as an initialization to help single-step attacks better solve the inner maximization problems.

Table 6.8: Performance of using Backward Smoothing alone on CIFAR-10 dataset using ResNet-18 model.

Method	Nat (%)	Rob (%)
Fast AT	84.79	46.30
Fast TRADES	84.80	46.25
Backward Smoothing Alone	69.87	39.26

Table 6.9: Performance of using Backward Smoothing alone on CIFAR-100 dataset using ResNet-18 model.

Method	Nat (%)	Rob (%)
Fast AT	60.35	24.64
Fast TRADES	60.22	19.40
Backward Smoothing Alone	43.47	18.51

6.7.2 More Experiments for Backward Smoothing using Multiple Random Points

We also conducted extra experiments using multiple random points for the Backward Smoothing method. As can be seen from Table 6.10, a single random point already leads to similar performance as multiple random points but saves more time. Note that our target is to improve the efficiency of adversarial training, therefore, we only use a single random point for randomized smoothing in our proposed method.

Table 6.10: Sensitivity analysis on the number of random points used in Backward Smoothing on the CIFAR-10 dataset using ResNet-18 model.

# RandPoints	Rob (%)	Time (m)
1	52.50	164
2	52.67	204
5	52.70	316
10	52.73	510

Table 6.11: Sensitivity analysis of γ on the CIFAR-10 and CIFAR-100 datasets using ResNet-18 model.

Dataset	CIFAR-10		CIFAR-100	
γ	Nat (%)	Rob (%)	Nat (%)	Rob (%)
0.1	82.43	52.13	56.62	29.34
0.5	82.53	52.34	56.95	29.85
1.0	82.38	**52.50**	56.96	**30.50**
2.0	82.29	52.42	56.16	29.88
5.0	81.50	52.32	56.10	429.83

6.7.3 Ablation Studies

We also perform a set of ablation studies to provide a more in-depth analysis on Backward Smoothing.

Effect of γ We analyze the effect of γ in Backward Smoothing by fixing β and the attack step size. Table 6.11 summarizes the results. In general, γ does not have a significant effect on the final model robustness; however, using too large or too small γ would lead to slightly worse robustness. Empirically, $\gamma = 1$ achieves the best performance on both datasets.

Table 6.12: Sensitivity analysis of β on CIFAR-10 and CIFAR-100 datasets using ResNet-18 model.

Dataset	CIFAR-10		CIFAR-100	
β	Nat (%)	Rob (%)	Nat (%)	Rob (%)
2.0	84.87	46.46	62.22	24.83
4.0	84.58	50.01	59.03	27.58
6.0	83.96	51.65	57.46	28.66
8.0	82.48	51.88	57.51	29.38
10.0	82.38	**52.50**	56.96	**30.50**
12.0	81.63	52.38	56.46	29.95

The Effect of β We conduct the ablation studies to figure out the effect of β in the Backward Smoothing method by fixing γ and the attack step size. Table 6.12 shows the experimental results. Similar to what β does in TRADES [ZYJ$^+$19], here in Backward Smoothing, β still controls the trade-off between natural accuracy and robust accuracy. We observe that with a larger β, natural accuracy keeps decreasing and the best robustness is obtained with $\beta = 10.0$.

6.7.4 Experiments on different perturbation strength

We also conducted experiments to compare the performance in other ϵ settings. Specifically, we compare the $\epsilon = 4/255$ case and $\epsilon = 12/255$ case in Table 6.13 and 6.14. Both tables again show the advantages of our Backward Smoothing algorithm over other baselines.

6.7.5 PGD based Backward Smoothing

We also wonder whether Backward Smoothing is compatible with Adversarial Training, *i.e.*, can we use a similar initialization strategy for improving Fast AT? Following the same idea

Table 6.13: Performance comparison on CIFAR-10 using ResNet-18 model ($\epsilon = 4/255$).

Method	Nat (%)	Rob (%)	Time (m)
AT	88.43	68.85	428
Fast AT	89.40	65.80	**90**
Fast AT (2-step)	**89.50**	66.89	129
Fast AT (GradAlign)	89.15	65.78	401
TRADES	88.35	**70.05**	478
Fast TRADES	89.20	66.71	136
Fast TRADES (2-step)	88.73	67.86	174
Backward Smoothing	87.22	69.67	165

Table 6.14: Performance comparison on CIFAR-10 using ResNet-18 model ($\epsilon = 12/255$).

Method	Nat (%)	Rob (%)	Time (m)
AT	72.92	**39.56**	433
Fast AT	64.06	26.14	**90**
Fast AT (2-step)	77.95	33.68	127
Fast AT (GradAlign)	76.02	33.03	400
TRADES	**76.07**	39.11	475
Fast TRADES	64.12	25.93	137
Fast TRADES (2-step)	75.98	29.91	174
Backward Smoothing	71.90	35.22	166

as in Section 6.4, we tend to find an initialization $\boldsymbol{\xi}^*$ such that

$$\boldsymbol{\xi}^* = \operatorname*{argmin}_{\boldsymbol{\xi} \in \mathcal{B}_\epsilon(\mathbf{0})} ([f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\xi}) - \gamma \boldsymbol{\psi}]_y)^2,$$

where $\boldsymbol{\psi}$ is the random vector and we only take the y-logit since the CrossEntropy loss used in adversarial training mainly cares about the y-logit. We test this on CIFAR-10 using ResNet-18 model, and summarize the results in Table 6.15. We can observe that combining Backward Smoothing with PGD can still achieve certain level of improvements but not as good as when combined with TRADES.

6.7.6 Comparison of Backward Smoothing and the ODI attack

We notice that [TSE20] proposed an ODI attack which shares some similarity as our proposed Backward Smoothing method (it also computes an initialization direction before normal attack), however, they are quite different in several ways. In this subsection, we compare the differences and argue that our method is not like using ODI attack for adversarial training.

First and foremost, the formulation is totally different. In ODI attack, its initialization is solved by

$$\max_{\boldsymbol{\xi}\in\mathcal{B}_\epsilon(\mathbf{0})} f_{\boldsymbol{\theta}}(\mathbf{x})^\top \boldsymbol{\psi},$$

where $\boldsymbol{\psi} \sim U(-1,1)$. This is totally different formulation compared to (6.4.4) for Backward Smoothing. Second, the motivations are different, ODI is a type of adversarial attack, which aims at lowering the prediction accuracy of target classifier, while our Backward Smoothing focuses on smoother the objective function. In fact, if one adopts Backward Smoothing for an attack, it can hardly achieve superior performances (smoother loss landscape also means hard to increase the loss significantly). To be more convincing, we also compare the result of applying ODI attack for adversarial training in Table 6.15. We can observe that using ODI attack for robust training achieves much worse robustness performances compared to Backward Smoothing both for PGD based and TRADES based strategies.

6.8 Conclusions

In this paper, we analyze the reason why single-step robust training without random initialization would fail and propose a new understanding towards Fast Adversarial Training by viewing random initialization as performing randomized smoothing for the inner maximization problem. Following this new perspective, we further propose a new initialization strategy, Backward Smoothing. The resulting method closes the robustness gap to state-of-the-art robust training methods and significantly improves model robustness over single-step

Table 6.15: Performance of single-step based robust training strategy on CIFAR-10 dataset using ResNet-18 model.

Method	Nat (%)	Rob (%)
Fast AT	84.79	46.30
Fast TRADES	84.80	46.25
Backward Smoothing (PGD)	82.69	47.96
Backward Smoothing (TRADES)	82.38	52.50
ODI (PGD)	85.20	43.41
ODI (TRADES)	84.83	49.37

robust training methods.

CHAPTER 7

Conclusions and Future work

In this chapter, we summarize the introduced works in previous chapters, discuss the pros and cons for them, and, based on them, hash out the blueprint of possible future directions.

In Chapter 2 and Chapter 3, we built better tools for evaluating model robustness under different settings (different levels of information access). In Chapter 2, we have built a Frank-Wolfe framework for efficient and effective adversarial attacks in white-box and black-box settings. In Chapter 3, we introduce the RayS attack, which focuses on evaluating model robustness in the hard-label setting. We also proposed a new robustness metric called average decision boundary distance, which can be used to detect "falsely robust" models.

In Chapter 4, we carefully examined the fundamental limit of robust training methods on image distributions and observed a large gap between the theoretical intrinsic robust limit and the best robustness achieved by state-of-the-art robust classifiers.

In Chapter 5, we carefully examine the common belief that network width helps adversarial robustness via thorough experiments and also careful analyses. And we have reached a count-intuitive conclusion that the increased network width helps natural generalization but may hurt robustness. To reach this conclusion, we conduct a thorough empirical study on the network width and adversarial robustness, and also investigate the underlying reasons behind the scenes.

In Chapter 6, we propose a new understanding towards Fast Adversarial Training [WRK20] by viewing random initialization as performing randomized smoothing for the inner maximization problem. We then show that the smoothing effect by random initialization is not enough under adversarial perturbation constraint. To address this issue, we propose a new initialization strategy, Backward Smoothing. The resulting method closes the robustness gap to state-of-the-art robust training methods and significantly improves model robustness over single-step robust training methods.

Despite all the progress, there are still many things that remain unclear for robust training, which requires further understandings and more in-depth study towards the adversarial robustness in deep learning. Here we summarize some possible future research directions in the following.

- **Adversarial Robustness for Physical Environments**: Traditional adversarial examples and corresponding defenses are established based on static data, e.g., pictures from the ImageNet dataset. These methods could fail in the physical environments due to natural transformations such as viewpoint shifts, lens distortions, folds in clothes and different gestures of animals. An adversarial perturbation that works on a static image or frame may not work for the same object in the real world. This largely restricts the practical impact of the current adversarial attack and defense algorithms. Therefore, it is of great interest to study robust adversarial attacks that constantly fool the neural network under various transformations and natural changes, as well as how to defend such kind of "persistent" attacks. My insight is to generate adversarial perturbations upon the worst-case natural transformations and rigorously formulate them using a max-min optimization problem, which can be solved via advanced nonconvex optimization techniques. In addition, I plan to emphasize such type of "persistent" attacks in defensive robust training to further improve model robustness under physical environments.

- **Adversarial Robustness on Discrete Data**: Discrete data, such as text, graphs, categorical features, pervasively exist in real life. Machine learning models dealing with such data can also be attacked by perturbations defined in discrete space. For example in fake news detection tasks, an attacker may mislead a detector by simply changing the word "pleased" into "delighted". In the representation learning task on graphs, an attacker may fool the graph embedding model to learn wrong representations for target nodes by injecting unnoticeable changes to the graph structure (deleting or adding edges). Therefore, studying the adversarial robustness of models training on discrete data is equally important as continuous ones but more challenging: finding the optimal discrete perturbation is intrinsically an NP-hard combinatorial optimization task; gradient-based methods cannot be directly applied in discrete space. It still remains an open problem how to perform effective and efficient searches towards the optimal solutions. My research [CG20] already showed that the efficient search strategy is feasible in continuous cases. To extend my research for discrete cases, the key is to properly define the perturbation distance and decision boundary in the discrete space and then build efficient search strategies upon them. I plan to develop efficient search strategies to systemically explore adversarial robustness and utilize the knowledge gained to further build robust models on discrete data.

- **Efficient Large Scale Adversarial Learning**: From the record-high performances in ImageNet challenges to the success of pre-trained language models such as BERT, GPT-3, in the past decade, we have observed the huge leap in AI brought by large scale machine learning model training. However, such models usually need days or months of training time under moderate hardware conditions. Furthermore, it is almost computationally infeasible to perform adversarial training on those models to enhance their robustness. Therefore, it is of great value to make large scale robust model training more efficient. Achieving such a goal would require efforts from both robust training and deep learning optimization. My previous studies on optimizing deep neural network training via adaptive gradient methods [CZT$^+$20, ZCC$^+$20] have laid solid foundations for improving the training efficiency of large scale machine learning systems, however, they are universal optimization methods that work for general problems. Recent studies show that popular optimizers such as Adam work well on certain tasks (e.g., language modeling) but have degraded performance on the others (e.g., image classifications). This suggests that specialized task-specific optimizers may achieve better efficiency compared to universal ones on certain tasks. Combining with my proposed efficient robust training strategy (Chapter 6), I aim to design task-specific efficient optimizers that allow efficient adversarial training for widely used large scale machine learning models.

Back to the main subject of this dissertation, the current robust training method and the corresponding robust models are still far from satisfactory. There still exists a large gap between the current best robust models with the theoretical limit of adversarial robustness. It can also be anticipated that as we getting deeper and deeper understandings towards robust training, more and more problems will be brought up and further solved by better robust models. Hopefully the works we introduced in this dissertation can help researchers in this area better evaluate and understand the fundamentals of robust training and actually contribute to the development of better robust models.

Bibliography

[ACFH19] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2019.

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

[ADO20] Abdullah Al-Dujaili and Una-May O'Reilly. Sign bits are all you need for black-box attacks. In *International Conference on Learning Representations*, 2020.

[AF20] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 2020.

[AUH$^+$19] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *NeurIPS*, pages 12214–12223, 2019.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252, 2019.

[BCM19] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower bounds on adversarial robustness from optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[BCS$^+$16] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4945–4949. IEEE, 2016.

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[BG18] Krishnakumar Balasubramanian and Saeed Ghadimi. Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates. *arXiv preprint arXiv:1809.06474*, 2018.

[BHLS17] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*, 2017.

[BLPR19] Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya P. Razenshteyn. Adversarial examples from computational constraints. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 831–840. PMLR, 2019.

[Bor75] Christer Borell. The Brunn-Minkowski inequality in Gauss space. *Inventiones mathematicae*, 30(2):207–216, 1975.

[BRB18] 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.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[CAD$^+$18] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[CBG$^+$17] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann N. Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In Doina Precup and Yee Whye Teh, editors, *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863. PMLR, 2017.

[CG19] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10836–10846, 2019.

[CG20] Jinghui Chen and Quanquan Gu. Rays: A ray searching method for hard-label adversarial attack. In *Proceedings of the 26rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[CH20a] F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020.

[CH20b] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[CJW19] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. *arXiv preprint arXiv:1904.02144*, 3, 2019.

[CLC$^+$19] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[CMV$^+$16] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, pages 513–530, 2016.

[CRK19] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, pages 1310–1320, 2019.

[CRS$^+$19] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, pages 11192–11203, 2019.

[CSC$^+$20] Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *International Conference on Learning Representations*, 2020.

[CSZ$^+$17] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017.

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[CYZ$^+$18] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*, 2018.

[CZC$^+$17] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Show-and-fool: Crafting adversarial examples for neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017.

[CZS$^+$17] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[CZT$^+$20] Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *International Joint Conferences on Artificial Intelligence*, 2020.

[CZYG20] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. A frank-wolfe framework for efficient and effective adversarial attacks. In *AAAI*, 2020.

[DAL$^+$18] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *International Conference on Learning Representations*, 2018.

[DBW12] John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[DDS$^+$09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[DLL$^+$19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685, 2019.

[DLP+18] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[DMM18] Dimitrios Diochnos, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarial risk and robustness: General definitions and implications for the uniform distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[Doh19] Elvis Dohmatob. Generalized no free lunch theorem for adversarial robustness. In *International Conference on Machine Learning (ICML)*, 2019.

[FFF18a] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[FFF18b] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.*, 107(3):481–508, 2018.

[FKM05] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics, 2005.

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[GCL+19] Ruiqi Gao, Tianle Cai, Haochuan Li, Cho-Jui Hsieh, Liwei Wang, and Jason D Lee. Convergence of adversarial training in overparametrized neural networks. In *Advances in Neural Information Processing Systems*, pages 13029–13040, 2019.

[GDS$^+$19] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *International Conference on Computer Vision (ICCV)*, 2019.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[GMF$^+$18] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[GQU$^+$20] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020.

[GR15] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[GRCVDM18] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *International Conference on Learning Representations*, 2018.

[GSS15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[HA17] 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.

[HDY$^+$12] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[HLM19] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *ICML*, pages 2712–2721, 2019.

[HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[HT17] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. *arXiv preprint arXiv:1702.05983*, 2017.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[IEA$^+$18] Andrew Ilyas, Logan Engstrom, Anish Athalye, Jessy Lin, Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Black-box adversarial attacks with limited queries and information. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[IEM19] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *International Conference on Learning Representations*, 2019.

[IST$^+$19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 125–136, 2019.

[Jag13] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[JBZB19] Jörn-Henrik Jacobsen, Jens Behrmann, Richard S. Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *ICLR*. OpenReview.net, 2019.

[JHG18] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 8580–8589, 2018.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[KBD$^+$17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 2017.

[KGB16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[KGB17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*. OpenReview.net, 2017.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KW20] Jungeum Kim and Xiao Wang. Sensible adversarial learning, 2020.

[LAG+19] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *SP*, pages 656–672. IEEE, 2019.

[LBB+98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010.

[LCLS18] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *International Conference on Data Mining (ICDM)*, 2018.

[LeC98] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[LJ16] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[LLD$^+$18] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[LLW$^+$19] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning*, pages 3866–3876, 2019.

[LSL$^+$20] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *arXiv preprint arXiv:2006.08403*, 2020.

[LWJC20] Bai Li, Shiqi Wang, Suman Jana, and Lawrence Carin. Towards understanding fast adversarial training. *arXiv preprint arXiv:2006.03089*, 2020.

[MAS19] Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *International Conference on Machine Learning*, pages 4636–4645, 2019.

[MC17] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[MDH$^+$12] Abdel-rahman Mohamed, George E Dahl, Geoffrey Hinton, et al. Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech & Language Processing*, 20(1):14–22, 2012.

[MDM19] Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *AAAI Conference on Artificial Intelligence*, 2019.

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582. IEEE Computer Society, 2016.

[MLW$^+$18] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *International Conference on Learning Representations*, 2018.

[MMS$^+$18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[MNA$^+$17] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[MZME19] Saeed Mahloujifar, Xiao Zhang, Mohammad Mahmoody, and David Evans. Empirically measuring concentration: Fundamental limits on intrinsic robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[Nak19] Preetum Nakkiran. Adversarial robustness may be at odds with simplicity. *CoRR*, abs/1901.00532, 2019.

[NMKM19] Amir Najafi, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. Robustness to adversarial perturbations in learning from incomplete data. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 5542–5552, 2019.

[OOS17] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning (ICML)*, 2017.

[PMG16] 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.

[PMG$^+$17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[PMJ$^+$16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[PMW$^+$16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[PTL+18] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[PYD+20] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Hang Su, and Jun Zhu. Boosting adversarial training with hypersphere embedding. *CoRR*, abs/2002.08619, 2020.

[PYD+21] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2021.

[PŽ06] Remigijus Paulavičius and Julius Žilinskas. Analysis of different norms and corresponding lipschitz constants for global optimization. *Technological and Economic Development of Economy*, 12(4):301–306, 2006.

[QMG+19] 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 *NeurIPS*, pages 13847–13856, 2019.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[RSPS16] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 1244–1251. IEEE, 2016.

[RWK20a] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. *ICML*, 2020.

[RWK20b] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. *CoRR*, abs/2002.11569, 2020.

[RXY+19] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial training can hurt generalization. *CoRR*, abs/1906.06032, 2019.

[RXY+20] 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.

[SABB20] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and R Venkatesh Babu. Guided adversarial attack for evaluating and enhancing adversarial defenses. *arXiv preprint arXiv:2011.14969*, 2020.

[Sha17] Ohad Shamir. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research*, 18(52):1–11, 2017.

[SHC+17] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever.

Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[SHK12] Ilya Sutskever, Geoffrey E Hinton, and A Krizhevsky. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[SHS$^+$19] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations (ICLR)*, 2019.

[SJ17] Matthew Staib and Stefanie Jegelka. Distributionally robust deep learning as a generalization of adversarial training. *Machine Learning and Computer Security Workshop*, 2017.

[SKC18] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*, 2018.

[SKN$^+$18] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *International Conference on Learning Representations*, 2018.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[SLR$^+$19] 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 *NeurIPS*, pages 11292–11303, 2019.

[Smi17] Leslie N Smith. Cyclical learning rates for training neural networks. In *WACV*, pages 464–472. IEEE, 2017.

[SND18] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. *International Conference on Learning Representations*, 2018.

[SNG+19] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, pages 3358–3369, 2019.

[SST+18] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 5019–5031, 2018.

[ST78] Vladimir N Sudakov and Boris S Tsirelson. Extremal properties of half-spaces for spherically invariant measures. *Journal of Soviet Mathematics*, 9(1):9–18, 1978.

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[SWMJ20] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. On pruning adversarially robust neural networks. *CoRR*, abs/2002.10509, 2020.

[SZS$^+$13] 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.

[SZS$^+$14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2014.

[TCBM20] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

[TPG$^+$17] Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *CoRR*, abs/1704.03453, 2017.

[TSE$^+$19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*. OpenReview.net, 2019.

[TSE20] Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white-and black-box attacks. *Advances in Neural Information Processing Systems*, 33, 2020.

[TTC$^+$18] Chun-Chen Tu, Pai-Shun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. *CoRR*, abs/1805.11770, 2018.

[TXT19] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019.

[UOKO18] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*, pages 5025–5034, 2018.

[WCAJ18] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mix-Train: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

[WCC$^+$20] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *arXiv preprint arXiv:2010.01279*, 2020.

[WK18] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018.

[WMB$^+$19] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, pages 6586–6595, 2019.

[WRK20] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

[WSG$^+$14] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[WSMK18] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[WXW20] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

[WZC$^+$18] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018.

[WZY$^+$20] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*. OpenReview.net, 2020.

[XCL$^+$17] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darell, and Dawn Song. Can you fool ai with adversarial examples on a visual turing test? *arXiv preprint arXiv:1709.08693*, 2017.

[XWZ$^+$18] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *International Conference on Learning Representations*, 2018.

[YRZ$^+$20] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in Neural Information Processing Systems*, 33, 2020.

[YZS17] Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Generalized conditional gradient for sparse estimation. *The Journal of Machine Learning Research*, 18(1):5279–5324, 2017.

[ZCC$^+$20] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *NeurIPS Workshop on Optimization for Machine Learning*, 2020.

[ZCGE20] 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.

[ZCZG19] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *Machine Learning Journal*, 2019.

[ZCZG20] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *BMVC*. BMVA Press, 2016.

[ZW19] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *Advances in Neural Information Processing Systems*, pages 1829–1839, 2019.

[ZX20] Haichao Zhang and Wei Xu. Adversarial interpolation training: A simple approach for improving model robustness, 2020.

[ZXH$^+$20a] 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. In *International Conference on Machine Learning*, pages 11278–11287. PMLR, 2020.

[ZXH$^+$20b] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. *CoRR*, abs/2002.11242, 2020.

[ZYJ$^+$19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482, 2019.