ON THE FEATURE ALIGNMENT OF DEEP VISION MODELS
Explainability and Robustness Connected At Hip

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Department of Electrical and Computer Engineering

Zifan Wang

M.S. in Electrical and Computer Engineering, Carnegie Mellon University
B.S. in Electrical and Information Technology, Beijing Institute of Technology

Carnegie Mellon University
Pittsburgh, PA

August 2023

Acknowledgements

One of my most deeply held beliefs is that we become the sum of our choices. I am profoundly grateful for the decisions that have led me to this point, together with the lights on my achievements and the hardest moments in the shadow. Supports that I received from my family, friends, advisors and my beloved cat Pikachu will never be forgotten. Here, I would like to acknowledge those who have played significant roles and perhaps made lifelong impacts on my journey, which began in the southwest of China and continues with a destination yet to be determined.

It usually won't make a kid popular to talk physics, space, and time but that was a type I would be categorized into when I was in my high-school age. These topics were brought to me in *the History of Time* (written by Stephen Hawking as educational reading instead of a textbook) and they became the roots for my interests in theories and technologies. Nonetheless, during my undergraduate years, I found myself drawn to debate competitions — perhaps a serendipitous distraction in hindsight. This was largely due to the allure of critical thinking and the philosophy of logic, which resonated deeply with a curious student like myself. I owe much to teammates such as Tianchao Wang, Lingyu Zhao, Menhui Tang, and others, for it was during those debates that I honed my ability to reason and present arguments based on evidence and logical thought. My time on the debate team has undeniably become the bedrock of my research methodology, continuing to influence me to the present.

I wasn't sure what I would feel passionate to commit to after finishing my undergraduate course with a mixture of knowledge in electronics, signals, programming, and artificial intelligence. Fortunately, I secured a few chances to travel to and study at the University of Alberta, the National University of Singapore and the Hong Kong University of Science and Technology where I was given chances to participate in several projects related to machine learning, computer graphics, and robotics. I was able to see how machine learning could possibly reshape the future with the striking news from AlphaGo and one of the top ones in Computer Science, Carnegie Mellon University (CMU), has offered me a chance to build that future together with many talented individuals.

Anupam Datta delivered the first lecture I attended at the Silicon Valley campus of CMU, delving into the intricate mechanics of deep neural networks and elucidating them through influence-directed explanations. That lecture was, in many ways, a turning point for me, solidifying my decision to embark on research by pursuing a Ph.D. Notably, Anupam later became my Ph.D. advisor, as the realm of explainability in deep learning deeply intrigued me. I never anticipated the good fortune of having Matt Fredrikson co-advise me, someone endowed with extensive knowledge and experience in ML security, privacy, adversarial robustness, and explainability. Much of my work bridging ML explainability and

robustness can be attributed to their efforts in providing me with a great mentorship. During my Ph.D. I was able to explore many exciting yet open problems in deep learning with colleagues in CMU and outside collaborators, including Klas Leino, Kaiji (Caleb) Lu, Ravi Mangal, Emily Black, Han Zhang, Saranya Vijayakumar, Kai Hu, Jiaming (Andy) Zou, Chi Zhang, Yuhang, Yao, Haofan Wang, Clement Fung, Piotr Mardziel, Shayak Sen, Carlee Joe-Wong, Corina Pasareanu, J. Zico Kolter, Somesh Jha, Vijay Ganesh, Zi Wang, Nan Ding, Tomer Levinboim, Xi Chen, Radu Soricut, Garima, Pradyumna Narayana and others. In particular, I want to give special thanks to my committee members, Anupam Datta, Matt Fredrikson, Lujo Bauer, Sugato Basu and David Alvarez-Melis, for providing comments and feedback for the thesis and my defense.

The works completed in this thesis were developed with the support of NSF grant CNS-1704845 as well as by DARPA GARD Contract HR00112020006 and support from ARO-MURI AWD00001188, "Cohesive and Robust Human-Bot Cybersecurity Teams". The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of DARPA, the Air Force Research Laboratory, the National Science Foundation, or the U.S. Government. I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V and Titan Xp GPU used for some works in this thesis. My work done during my internship at Google was supported by Google's GPU/TPU compute clusters.

It was never an easy journey to obtain a Ph.D. in a foreign country with a foreign language, especially with no company of family and the effort of building a fresh new interpersonal network. COVID-related restrictions and visa issues have significantly affected my entire Ph.D. journey, preventing me from traveling back to China since 2019.The supports from my parents, Maoliang Wang and Ying Zhang, are distant but never weak and I feel grateful for that. During my Ph.D., I received invaluable support from Yi Yue, Jingyuan Luo, Tao Sang, Sizhi Zhou, and numerous friends in China (I look forward to seeing them again). Additionally, Yangfan Ran, Xing Zhang, Zhipeng Bao, and many friends in the States stood by me. Their support was especially crucial during the global COVID lockdown.

Finally, I feel grateful for all support and guidance I have received. I wish all the best for my family and friends. Our paths have crossed at the moment and they will cross again in the future of all possibilities.

Abstract

Deep Neural Networks (DNNs) have recently demonstrated remarkable performance that is comparable to humans. However, these models pose a challenge when it comes to answering whether their behaviors, ethical values, and morality always align with humans' interests. This issue is known as *(mis)alignment* of intelligent systems. One basic requirement for deep classifiers to be considered aligned is that their output is always semantically equivalent to that of a human, who possesses the necessary knowledge and tools to solve the problem at hand. Unfortunately, verifying the alignment between models and humans on outputs is often not feasible, as it would be impractical to test every sample from the distribution.

A lack of output alignment of DNNs has been evidenced by their vulnerability to adversarial noise, which are unlikely to affect a human's response. This weakness originates from the fact that important features used by the model may not be semantically meaningful from a human perspective, an issue which we will term as *feature (mis)alignment* in vision tasks. Being (perceptually) aligned with humans on useful features is necessary to preserve output alignment. Thus, the goal of this thesis is to evaluate and enhance the feature alignment of deep vision classifiers to promote output alignment.

To evaluate feature alignment, we introduce *locality*, a metric based on *explainability* tools that guarantee faithful returns of important features contributing towards the models' outputs. Consequently, the first contribution of the thesis shows that modern architectures, e.g., Vision Transformers (ViTs), the state-of-the-art classifiers on many tasks, are misaligned in features. Our second contribution, on the other hand, shows that improved *adversarial robustness* leads to improved locality. To be specific, we find that a robust model has better locality than any non-robust model and the locality of a model increases as it becomes more robust. Inspired by this finding, our third contribution is to improve robustness with a novel technique, TrH regularization, based on a direct minimization of PAC-Bayesian generalization bound for robustness. Our technique provides the new *state-of-the-art* robustness for ViTs. However, as robustness is often measured by running existing attacks, the guarantee is only *empirical* and may fail against adaptive attacks. The last contribution of this thesis introduces GloRo Nets, which entail a built-in formal robustness verification layer based on the global Lipschitz constant of the model. Unlike a probabilistic guarantee provided by Randomized Smoothing, GloRo Nets have a *deterministic* guarantee and significantly improve the state-of-the-art *provable* robustness under ℓ_2-norm-bounded threats.

Robustness is necessary for feature alignment but is probably not sufficient, as there are many other unspecified requirements that would result in misalignment. In conclusion, the thesis discusses the issue of under-specification in classification and its connection to alignment, together with potential remedies for addressing the issue as another step towards feature alignment in deep learning.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep Neural Networks (DNNs) have successfully revolutionised the ways how humans interact with the natural world. By effectively learning complex representations from vast amounts of data, DNNs have demonstrates *human-level* performance over thousands of vision and language tasks. However, despite their impressive learning capabilities, the neural reasoning behind their decision-making processes still remains opaque to humans. As a result, analyzing their behavior and verifying their performance can be a challenging task. The lack of DNN transparency places a crucial question to us,

Is the purpose put into the nueral network the purpose that we really desire?

—- *Norbert Wiener (1960)*

This is the concern raised by Norbert Wiener, pioneer mathematician in theorizing Artificial Intelligence (AI), in 1960s. The question above opens up research fields that is often referred to as AI *alignment*. Namely, an aligned AI system advances the intended objective; a misaligned AI system is competent at advancing some objective, but not the intended one (Russell, 2010). The research of alignment includes but not limits to align the model's ethical value, morality, and desired behavior with humans' interests and finally benefit us (Gab; Brundage et al., 2020; Christiano et al., 2017; Muggleton & Chater, 2021; Ouyang et al., 2022).

For vision tasks interested in this thesis, we dive deep into a basic building block for other higher-level requirements of *alignment* – that is, for deep classifiers to be considered aligned their outputs should be semantically equivalent to those of a human, who has the necessary knowledge and tools to solve the problem at hand. Unfortunately, verifying the alignment between models and humans on outputs, which we term as *output alignment*, is often not feasible, as it would be impractical to test every sample from the data distribution.

Notice that the term *alignment* or *human alignment* can also refer to the social impacts of artificial intelligence systems, which require AI to generate content aligning with human values and intentions. As AI systems become more capable, there's a growing concern about ensuring they behave in ways that are beneficial, safe, and in line with human intentions. Reinforcement Learning with Human Feedback (RLHF) (Brown et al., 2020; Christiano et al., 2017) and constitutional AI (Bai et al., 2022), for example, are popular means to realize *human alignment* for language models. However, when the term output alignment is used in this thesis, we particularly refer to models that agree with humans on the outputs and do not require such outputs to be ethical.

Various researchers have demonstrated a lack of output alignment in deep classifiers, as evidenced by its vulnerability to adversarial noise in the input, which is unlikely to affect a human's response. For example, several studies have shown that models with seemingly good performance on the ImageNet (Deng et al., 2009) test set can barely be accurate on well-crafted ones (Carlini & Wagner, 2017c; Croce & Hein, 2020b; Goodfellow et al., 2015b; Madry et al., 2018a) and natural ones (Hendrycks et al., 2021), where humans do not find them confusing and give consistent responses compared to benign inputs. For safety-critical applications, what's even worse than knowing that the deployed model lacks output alignment and may fail is the uncertainty about when the catastrophic failure will occur.

One reason to blame for this weakness of DNN is that important features used by the model may not be semantically meaningful from a human perspective, an issue which we will term as *feature (mis)alignment* in vision tasks. Consider an example shown in Figure 1.1, where a human teaches a DNN to associate an image of cat to a label `cat`. Upon correctly learning the mapping from the training input to its label, does the human know which logic the model learns from the data? That is, is the correct classification based on cat-relevant pixels, e.g. face and eyes, which a human would use for describing cats, or those irrelevant pixels on the background? Put it simple, it is necessary for a model's output to align with a human's response to have the important features used by the model to align with truly important ones from a human's perception.

The aim of this thesis is to assess *feature alignment* of deep models and develop guides and tools to build well-aligned models. The beginning of this thesis, therefore, focuses on the following question,

How does one assess if a DNN uses truly useful features that humans would use to classify the same image?

First, to address this problem, we rely on a metric that leverages *feature attribution*, a family of *explainability* techniques with a faithful guarantee to return important features that contribute to model's decision. To efficiently check for whether these features are semantically meaningful to humans, we leverage bounding boxes in vision datasets, e.g. ImageNet (Deng et al., 2009). By comparing attributions with bounding

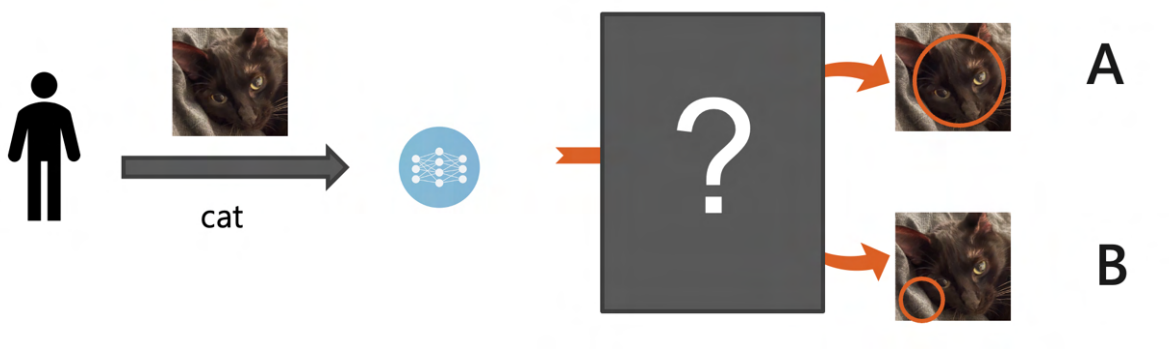


Figure 1.1: A human teaches a DNN to associate an image of cat to the label `cat` but the human is unsure about which way the classification is run inside the model: using cat-relevant pixels (logic A) or using cat-irrelevant pixels (logic B).

boxes, we introduce *locality*, a quantitative metric for measuring the degree of feature alignment. Second, our empirical evaluations on locality show that the often used metric, i.e. classification accuracy, is not a good indicator for locality, while *adversarially robust* models preserve higher locality than the standard (i.e. non-robust) ones. This connection we find between robustness and locality motivates our third contribution, which contains a set of techniques to improve model robustness. Finally, we highlight a limitation of robustness for realizing well-aligned models. Namely, for images with no truly useful features, a human-like behavior is to abstain from classification. Our contribution here is a new type of neural network that opts to abstain for unseen classes, moving another step further towards feature alignment.

The presentation of the thesis is organized as follows. Section 1.1 provides an overview of the way we define and measure *locality* using *feature attributions*. We find accurate models do not always have high locality. Chapter 2 expands the content in Section 1.1 with greater details. Section 1.2 summarizes the concept of *adversarial robustness* and its connection to model locality, with more details in Chapter 3. Section 1.3/1.4 focuses on a novel technique on improving the empirical/provable robustness and details to follow in Chapter 4/5. Lastly, Section 1.5 highlights that a limitation of robustness, i.e. robust models can classify white noise into a meaningful class (e.g. “dog") because conventional networks do not abstain for invalid inputs. The remedy proposed here is to a novel architecture that is self-explainable and opts to abstain by construction. We will cover greater details in Chapter 6.

1.1 Assessing feature alignment with Locality

The objective of this thesis is to evaluate and improve the alignment between the outputs generated by vision models and those produced by humans on visual classification tasks. Exhausting all samples from an unknown true distribution of data is not a feasible option for practical reasons. Thus, we instead explore

a condition essential for models to consistently agree with human outputs. This condition, which we refer to as *feature alignment*, entails models relying on (perceptually[1]) relevant features that humans, provided with the necessary knowledge and appropriate tools, would similarly employ. It should be noted that while feature alignment alone may be insufficient to achieve output alignment due to the need for neural reasoning steps to mirror the thought processes of humans. However, feature alignment alone is likely to be necessary to output alignment in vision models as the model without using similar features does not mimic humans for sure. As a result, feature alignment is still useful as a way to flag potential output misalignment on unseen data. For example, Figure 1.1 illustrates how a network with logic B lacks feature alignment with humans, given that the presence of a cat is not logically associated with the background of a blanket or couch. Therefore, logic B is unlikely to produce an output-aligned model and is likely to fail when presented with unseen images of cats with different backgrounds.

The initial step in assessing feature alignment involves identifying input features that significantly influence the model's outcome. In machine learning *explainability*, this is commonly referred to as *feature* attribution, a technique that computes a score for each input feature as its importance to the model's output. Care must be taken that the significance of a feature can vary substantially depending on the function segment being examined. Specifically, a function's output can display monotonic growth as a feature increases within a small neighborhood, but conversely, it may decline in a larger one. Therefore, to determine the importance of a feature, one must simultaneously specify the neighborhood of interest. In this thesis, we are interested in three types of neighborhood, i.e. (1) point-wise; (2) global; and (3) local neighborhoods, with the objective of identifying features that are point-wise important, globally important, and locally important and which contribute to the model's output. Among existing feature attributions (Binder et al., 2016; BOHNENBLUST et al., 1952; Erion et al., 2021; Fong & Vedaldi, 2017; Ghalebikesabi et al., 2021; Leino et al., 2018; Lundberg & Lee, 2017; Pan et al., 2021; Petsiuk et al., 2018; Ribeiro et al., 2016; Selvaraju et al., 2019; Shrikumar et al., 2017; Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017; Wang et al., 2020a, 2022), we choose Saliency Map (SM) (Simonyan et al., 2013) for the point-wise neighborhood and Integrated Gradient (IG) (Sundararajan et al., 2017) for the global neighborhood, provided that they are axiomatically justified to return scores that are faithful to the underlying model. For a local neighborhood, we include Boundary-based Integrated Gradient (BIG) from our recent work (Wang et al., 2022) for the same reason of faithfulness. To summarize, we find the important features that contribute most to the model's output with feature attributions, i.e. SM, IG and BIG, to complete our first step for evaluating feature alignment.

Given tools to discover the model's use of features, for the second step, we intend to evaluate whether

[1] as we are focusing on vision models in this thesis.

these highlighted features are truly useful. That is, *"do important features from the model's perspective perceptibly meaningful to human experts as well ?"* Typically, user studies involving a group of human volunteers are conducted to address such inquiries. In this thesis, we do not conduct such a survey because of a set of potential issues that may undercut the value of human feedback. More importantly, the only information that is relevant to our evaluation is a set of features that humans find perceptibly meaningful. This information is already present by human-labeled bounding boxes in commonly used vision datasets such as ImageNet (Deng et al., 2009). A bounding box typically comprises a vector of four elements denoting the coordinates of two diagonal vertices of a rectangle in the image, enabling the localization of the object. For classification, the model is expected to associate the output label with the object identified by the bounding box. Thus, the bounding box is an optimal choice in our use case to capture perceptibly meaningful features from humans' perspective.

We propose a set of locality metrics, akin to the conventional precision, recall, and F1-score used in binary classification, to assess the alignment between the model's important features identified through feature attribution and the perceptibly meaningful features identified through the bounding box. Using our locality evaluations, we find that despite the increasing test accuracy from 58.1% with a SqueezeNet (Iandola et al., 2016) to over 80% with a Vision Transformer (ViTs) (Dosovitskiy et al., 2020), their locality scores with SM, IG, and BIG are almost identical, respectively. In particular, the F1 locality scores are close to 0.5, indicating a weak alignment between feature attributions and the bounding box. This phenomenon is not surprising if visualizing feature attributions by overlaying importance scores over the image – attributions on the most accurate model are not very different from that on the worst model and they are both far from semantically meaningful to us.

Put together, with locality scores, the first conclusion drawn from this thesis is that *improved test is not a good indicator of improved feature alignment*. The findings we have in turn account for the well-known vulnerability of deep models against tiny and imperceptible noise (Goodfellow et al., 2015c) (as illustrated in Figure 1.2) – the real focus of the model is never the perceptibly meaningful ones so imperceptible and irrelevant noise can fool a model without getting caught by humans. For greater details, please refer to Chapter 2.

1.2 Better Locality by Improved Robustness

As evident by our experiments in Chapter 2, improving test accuracy is far from enough to improve the locality of the model. In this section, we show that *adversarial robustness* effectively improves locality, leading to a better-aligned model.

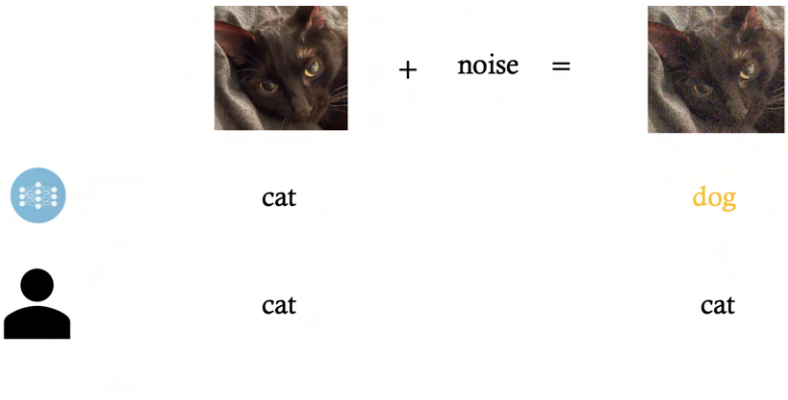


Figure 1.2: The well-known vulnerability of deep model against imperceptible noise, which is very unlikely to change a human's response to the image.

Figure 1.2 demonstrates how our human perception and a deep neural network differ in their sensitivity to perturbations. Despite the salt-and-pepper noise added to the original image, our brain can still recognize the cat, whereas the neural network changes its prediction accordingly, revealing its sensitivity to tiny noise. This sensitivity of a model to adversarial noise is known as *adversarial robustness*. Lack of adversarial robustness is indicative of a model's lack of feature alignment with humans. Hence, we pose the research question: *Can improved adversarial robustness lead to improved locality?*

To answer our question, we repeat the locality experiment for models that are trained with robustness-aware losses. Interestingly, the robust model, despite that it comes with a slightly lower test accuracy compared to the best ViT model we have in Section 1.1 (or Chapter 2), demonstrates a significant improvement on locality with all three attributions, i.e. SM, IG and BIG. Visualizations of these attributions on robust models also look much semantically meaningful and close to features in the bounding boxes. Put together, the second contribution of thesis is to reveal that *improved robustness effectively enhances the locality of the underlying model*.

On the benefit of robustness, we also show that gradient-based attributions become more similar to the normal vectors of the nearest/nearby decision boundaries, i.e. the most discriminating features that the model uses to make the decision at the input. We show a set of empirical results to validate this observation and derive a set of analytical results for justifying our findings.

The details discussed in this section are elaborated on in Chapter 3. The effectiveness of adversarial robustness in promoting locality and feature alignment motivates us to explore approaches for building more robust models in following chapters.

1.3 Training Robust Models Against Empirical Attacks

In order to build models that are robust against imperceptible perturbations to the inputs, we first consider to use small-norm-bounded perturbations to translate such imperceptibility. Formally, we consider an adversary with a budget of ϵ to inject any kind of noise to the input x but the perturbed input x' must be within the ϵ-ball of x w.r.t norm, $\|\cdot\|_p$, to make sure the perturbation is imperceptible to humans (otherwise we do not wish to make the model' predictions robust because humans may have already changed the responses to the perturbed input).

To date, there are a large body of attack methods designed for breaking deep models (Croce & Hein, 2020b; Goodfellow et al., 2015d; Madry et al., 2017; Nicolae et al., 2019). The common idea to defend against one of all of these attacks is to augment the training data with adversarial examples during training, which often results in a min-max optimization (Goodfellow et al., 2015d; Madry et al., 2017; Wong et al., 2020; Zhang et al., 2019c). That is, the inner-maximization tries to find a "worst" example that results in a great loss on classification with an attack and the outer-minimization hereby drives the weights of the model to minimize the loss on that "worst" example. Two well-known approaches follow this scheme are Adversarial Training (AT, Madry et al. (2017)) and TRADES (Zhang et al., 2019c).

Recently, Rice et al. (2020) observe a robust overfitting phenomenon, referred to as the *robust generalization gap*, in which a robustly-trained classifier shows much higher accuracy on adversarial examples from the training set, compared to lower accuracy on the test set. Indeed, several technical approaches have been developed that could alleviate this overfitting phenomenon, including ℓ_2 weight regularization, early stopping (Rice et al., 2020), label smoothing, data augmentation (Yun et al., 2019; Zhang et al., 2017), using synthetic data (Gowal et al., 2021) and etc.

According to learning theory, the phenomenon of overfitting can be characterized by a PAC-Bayesian bound (Alquier, 2021; Catoni, 2004; Germain et al., 2009; McAllester, 1999; Shawe-Taylor & Williamson, 1997) which upper-bounds the expected performance of a random classifier over the underlying data distribution by its performance on a finite set of training points plus some additional terms. Although several prior works (Gowal et al., 2021; Izmailov et al., 2018; Jin et al., 2022; Wu et al., 2020) have built upon insights from the PAC-Bayesian bound, none attempted to directly minimize the upper bound, likely due to the fact that the minimization of their forms of the PAC-Bayesian bound do not have an analytical solution.

In Chapter 4, we rely on a different form of the PAC-Bayesian bound (Germain et al., 2009), which can be readily optimized using a Gibbs distribution (Germain et al., 2016a) with which we derive a second-order upper bound over the robust test loss. Interestingly, the resulting bound consists of a regularization term

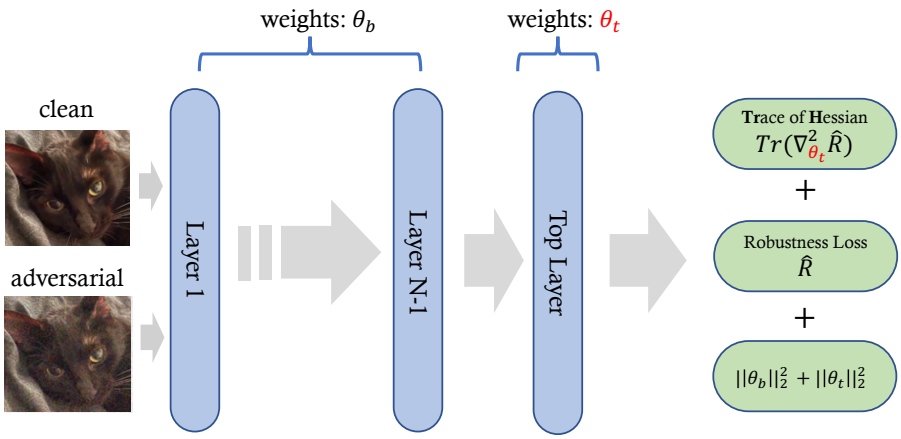


Figure 1.3: We propose Trace of Hessian (TrH) regularization for training adversarially robust models. In addition to an ordinary robust loss (e.g., TRADES (Zhang et al., 2019b)), we regularize the TrH of the loss with respect to the weights of the top layer to encourage flatness. The training objective is the result of direct PAC-Bayesian bound minimization in Chapter 4.

that involves *Trace of Hessian* (TrH) of the network weights, a well-known measure of the loss-surface flatness. For practical reasons, we limit TrH regularization to the top layer of the network only because computing a Hessian matrix and its trace for the entire network is too costly. We further derive the analytical expression of the top-layer TrH and show both theoretically and empirically that top-layer TrH regularization has a similar effect as regularizing the entire network. The resulting TrH regularization (illustrated in Figure 1.3) is less expensive and more memory efficient compared to other competitive methods (Gowal et al., 2021; Izmailov et al., 2018; Jin et al., 2022; Wu et al., 2020).

1.4 Training Robust Models With Formal Certification

One limitation of TrH regularization, together with other empirical defenses, against adversarial perturbations is that the robustness guarantee is only empirical w.r.t the attack used at validation time. That is, with attacks adaptive to these defenses, e.g. CW Nicolae et al. (2019), these defenses can still be vulnerable in the future. Thus, Chapter 5 hereby focuses on training methods that produce models whose robust predictions can be efficiently certified against adversarial perturbations. That is, the decision boundary in the input space is certified to be ϵ-away from the inputs as illustrated in Figure 5.

Over the last few years, a wide body of literature addressing robustness certification has emerged (Cohen et al., 2019a; Croce et al., 2019; Fromherz et al., 2021b; Huang et al., 2021b; Jordan et al., 2019b; Lee et al., 2020b; Leino et al., 2021d; Li et al., 2019a,b; Singla et al., 2022; Tjeng et al., 2019a; Trockman & Kolter, 2021; Wong & Kolter, 2018a). To date, the methods that achieve by far the best certified performance are derived

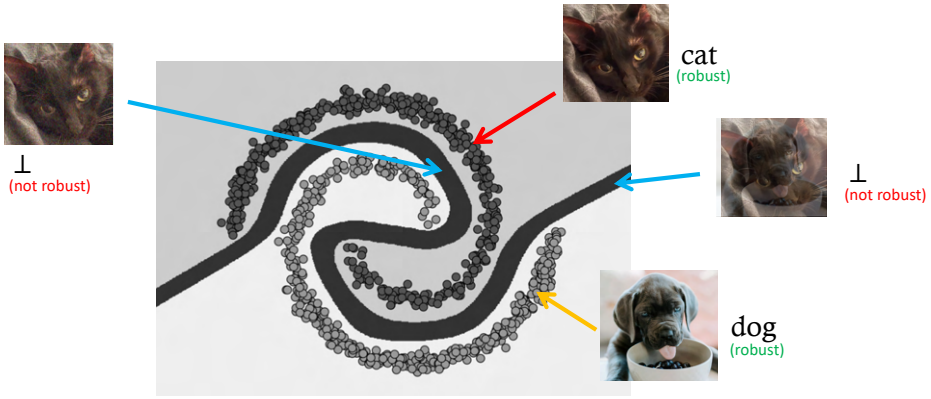


Figure 1.4: Plot of a certifiably robust decision boundary powered by our approach GloRo Nets. The thickness of $\perp$ stripe is at least 2ϵ so the model is certifiably ϵ-locally robust for both `dog` and `cat`. See Chapter 5 for the formal robust guarantee and details of GloRo Nets.

from *randomized smoothing* (Cohen et al., 2019a); however, this provides only a *probabilistic* guarantee—which may generate a false positive claim around 0.1% of the time (Cohen et al., 2019a)—and requires significant overhead for both evaluation and certification. By contrast, *deterministic* certification may be preferred in safety-critical applications, e.g., malware detection and autonomous driving.

Because of the highly non-linear boundaries learned by a neural network, deterministically certifying the robustness of its predictions usually requires specialized training procedures that regularize the network for efficient certification, as post-hoc certification is either too expensive (Katz et al., 2017; Sinha et al., 2018) or too imprecise (Fromherz et al., 2021b), particularly as the scale of the model being certified is increased. The most promising such approaches—in terms of both certified accuracy and efficiency—perform certification using Lipschitz bounds, meaning that the learning procedure must impose Lipschitz constraints on the layers during training, either through regularization (Leino et al., 2021d) or orthogonalization (Trockman & Kolter, 2021).

In Chapter 5, we propose a new type of neural network, Globally Robust (GloRo) Nets, with robustness certification by construction based on a *global* Lipschitz constant of the model (Figure **??**5Figure 5r approach to certification that shows promise in this regard uses Lipschitz bounds to efficiently calculate the robustness region around a point (Weng et al., 2018a; Zhang et al., 2018). In particular, when *global* bounds are used with GloRo Nets, it is possible to implement the bound computation as a neural network of comparable size to the original, making online certification nearly as efficient as inference. Unfortunately, current training methods do not produce models with sufficiently small global bounds for this to succeed Weng et al. (2018a). To address this issue, we introduce GloRo-CE, GloRo-TRADES and GloRo-EMMA, a set of losses that are both efficient in the run-time and memory and effective in obtaining a tight global bound

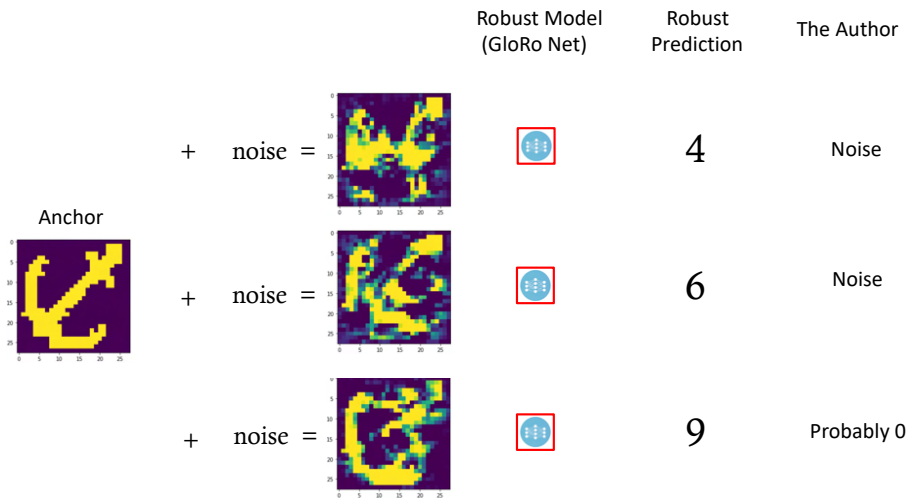


Figure 1.5: Plots of the insufficiency of robustness for realizing feature alignment. We add different noises to an image of `anchor` to make a robust model, i.e. GloRo Nets introduced in Chapter 5, to output robust predictions, which are unlikely to align with humans' outputs.

for certification purpose.

Focusing on the case of deterministic guarantees against ℓ_2-bounded perturbations, we show that this approach yields state-of-the-art verified-robust accuracy (VRA), while imposing little overhead during training and *none* during certification. For example, we find that we can achieve 29.2% VRA with a robustness radius of $\epsilon = 0.141$ on Tiny-ImageNet, surpassing all prior approaches by multiple percentage points. We are also the first to provide a deterministic robust guarantee on ImageNet, with a 14.2% VRA at radius $\epsilon = 1.0$. More importantly, to the best of our knowledge, this is the first time the deterministic robustness guarantee scales up to ImageNet, demonstrating the promise, facilitated by our architecture, of obtaining certifiably robust models in real-world applications.

1.5 Conclusions, Limitations and Future Work

In Chapter 6, we first discuss conclusions from our results in previous chapters. We conclude that to ensure vision models align with humans on feature use we need them to be resistant to imperceptible and adversarial noises, i.e. adversarial robustness. For that purpose, we introduce novel techniques and state-of-the-art performance.

Although robustness is essential for improving feature alignment, it is probably not sufficient on its own. Consider an example shown in Figure 1.5 where we take a robust model, i.e. a GloRo Net introduced in Chapter 5, trained on MNIST dataset and input a few perturbed images based on an image of anchor. These images are out of the distribution of digits 0 to 9 that the model is trained with; however, the model

gives robust predictions on these images as 4, 6 and 9. From the author's perspective, these images are probably just noises even the last image is probably similar to a digit 0 (but the robust model thinks it is 9).

It is not convincing to claim that the robust model shown in Figure 1.5 is a well-aligned model. The cause to the seemingly paradox between Figure 1.5 and our advocate of robustness from previous chapters is that robustness is necessary for the model to align on in-distribution images in features used by humans; however, for perturbed anchor images in Figure 1.5, they are out of the distribution that the model is trained with and they are not covered by the notion of *local robustness*, which GloRo Nets can certify. Geometrically, this is because GloRo Nets do not have a closed decision boundary so it fails to reject (i.e. by predicting $\bot$) for an out-of-distribution input.

In the rest part of Chapter 6, we dive into a potential solution to address the issue of a non-closing decision boundary in classification — *selective classification*. We consider this as a future direction for the work done in this thesis. We propose a novel idea of Centroid Nets, which is designed to reject two types of out-of-distribution points: the adversarial examples and the examples labeled with classes the model has not seen before. For a proof of concept, we provide theoretical results together with preliminary empirical results on Centroid Nets, showing how decision boundaries are closed compared to GloRo Nets.

Chapter 2

Assessing Feature Alignment with Locality

Feature Alignment is an essential property to realizing models that always align with humans on classification outputs, which is informally introduced in Chapter 2 as deep models must leverage features that are important to humans has been discussed. In this chapter, we provide a more concrete definition for feature alignment in Section 2.1, in addition to practical tools, i.e. feature attributions, and a metric, i.e. locality, that help in the practical assessment of feature alignment in Section 2.2 and 2.4.

2.1 Feature Alignment

A deep neural network is often abstracted as a function $f : \mathbb{R}^d \to \mathbb{R}^m$ where d, m are the dimensions of the input and output, respectively. This thesis particularly focuses on using deep neural networks to *classify* an input x into one of the m classes and we use $[m]$ to denote a closed set of integers between 0 and $m-1$. In this case, the j-th output neuron of $f(x)$ is referred to as *logit* score for class j, which can be normalized and projected by a softmax function into the probability space and interpreted as the conditional probability of the input belonging to class j, i.e. $Pr(j|x)$. In classification, we usually regard the class with the highest logit score as the network's prediction $F(x)$. Namely,

$$F(x) = \arg\max_{j \in [m]} f_j(x).$$

Here, we use the upper and lower cases to distinguish between the network's discrete integer prediction and continuous logit vector.

The goal of the learning problem is usually to find the mapping from one space to another, which in classification is a mapping from the feature space of the input to a space of probabilities or labels. In the set-up of supervised learning we consider in this thesis, we are given a set of pairs of inputs and labels, e.g. $D^n = \{(x, y)\}_{i=1}^n$, sampled from a true data distribution such that $(x, y) \sim \mathcal{D}$. Suppose $f^* : \mathbb{R}^d \to \mathbb{R}^m$ is

the true and unknown mapping that we desire, so f is what we have learned from D^n with some particular algorithm by minimizing the empirical risk.

In vision tasks, f^* abstracts the reasoning processes inherent to the human brain. This includes a sequence of steps where light, distance, and temporal information are converted into bio-electronic signals, combined, and then analyzed in our unique and intricate ways. While the inner workings of f^* remain enigmatic to us (arguably far more so than the neural net f), it suffices for our purposes to discern the significance of visual information in each image pixel based on its influence to f^*'s output.

A large body of works has focused on measuring the influence of input features of a specific set of functions, e.g. linear models, where game theory, causality and statistical importance play a major role for analyzing more general ones. In this thesis, we identify important pixels as those to which f^*'s output is notably *sensitive*. To describe this sensitivity, we introduce perturbations to the input and define a neighborhood based on these perturbations. The degree of sensitivity is then quantified through input-output correlations. We present Definition 1 to gauge the features that are truly significant to humans via f^*. Substituting f^* with f gives measures of feature importance for f.

Definition 1 (ρ-Important Feature) *Suppose $f^* : \mathbb{R}^d \to \mathbb{R}^m$ is the groundtruth mapping from the input x to the label y for any $(x, y) \sim \mathcal{D}$ in the data distribution. For the j-th input feature in x denoted as x_j, and a distribution $\mathcal{N}$, x_j is said to be ρ-important w.r.t $\mathcal{N}$ and y if*

$$\mathbb{E}_{\epsilon \sim \mathcal{N}}[(x_j + \epsilon) \cdot f^*_y(x_1, \cdots, x_j + \epsilon, \cdots, x_d)] \geq \rho,$$

where f^*_y is the y-th output of f^*.

The metric given in Definition 1 quantifies the correlation between a feature x_j and the function output f^* (or f) under perturbations governed by distribution $\mathcal{N}$. Features that exhibit strong correlations with the output are deemed important. The pivotal role of $\mathcal{N}$ is to delineate the specific subset of the input space — the targeted neighborhood — wherein the importance of x_j is being assessed. For instance, if $\mathcal{N}$ represents a narrow distribution centered at zero, such as a Delta distribution, ρ-importance corresponds to the function's *hyper-local* sensitivity towards the feature, akin to the concept of *influence* discussed by Ancona et al. (2018a); Leino et al. (2018) (highlighted by the green neighborhood in Figure 2.1). In the magnified view on the left, one could deduce that alterations to x_j within the green zone do not affect the output of the function, implying that x_j *does not hold significance for* f. Conversely, with a broader $\mathcal{N}$ distribution, the ρ-importance reflects either a *local* characteristic (illustrated by the yellow area) or a more *global* metric (as shown by the blue area) in Figure 2.1. Here, the yellow and blue arrows respectively indicate the positive and negative significance of x_j to the function. Therefore, the salient observation from

Figure 2.1: The importance of a feature x_j to the output of a function $f(x)$ depends on the neighborhood of interest. Arrows in different colors show the change of output with x_j increases in the corresponding regions in each color, respectively.

Figure 2.1 is imperative to clearly define the distribution of interest (or a reference neighborhood) to ensure unambiguous and consistent evaluations of input feature importance.

feature alignment. Evaluating if a significant feature x_j is pertinent to both the true mapping f^* and the learned function f enables us to discern the feature alignment between f and f^*. Nonetheless, it is essential to first ensure that the output scales of f and f^* are calibrated to make their importance metrics comparable. Such calibration can be achieved by transitioning from the logit output space to the probability output space, or through alternative calibration methodologies. For the purpose of this discussion, we presume that both f and f^* have undergone appropriate calibration. We here introduce *feature alignment* in Definition 2.

Definition 2 ((ρ, λ)-feature alignment) *Given a data distribution $\mathcal{D}$, a network $f : \mathbb{R}^d \to \mathbb{R}^m$, a noise distribution $\mathcal{N}$, a class of interest y, and a convex function $\mathsf{similarity}(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, define $\mathcal{I}_f(x)$ as a vector where*

$$\mathcal{I}_f^\rho(x)_j = x_j \cdot \mathbb{I}[x_j \text{ is } \rho\text{-important (w.r.t } \mathcal{N} \text{ and } y) \text{ to } f],$$

where $\mathbb{I}$ is an identity function and $\mathcal{I}_{f^*}^\rho(x)$ as a vector where

$$\mathcal{I}_{f^*}^\rho(x)_j = x_j \cdot \mathbb{I}[x_j \text{ is } \rho\text{-important (w.r.t } \mathcal{N} \text{ and } y) \text{ to } f^*].$$

Thus, f is said to be (ρ, λ)-aligned with f^* if

$$\mathbb{E}_{x \sim \mathcal{D}} \left[\mathsf{similarity}(\mathcal{I}_{f^*}^\rho(x), \mathcal{I}_f^\rho(x)) \right] \geq \lambda. \tag{2.1}$$

Objective	Math Definition	Empirical Approach
Important Features to DNNs	ρ-importance (Definition 1)	Feature Attribution (Section 2.2)
Important Features to Humans	ρ-importance (Definition 1)	Bounding Box (Section 2.3)
feature alignment	(ρ, λ)-feature alignment (Definition 2)	Locality (Section 2.4)

Table 2.1: An overview of relations between our definitions on feature alignment and empirical ways to measure them in Section 2.2, 2.3 and 2.4.

Definition 2 is a sound but perhaps practically infeasible to measure because (1) computing $\mathcal{I}_f^\rho(x)_j$ needs to visit every point in the perturbation neighborhood, which can be empirically approximated but expensive to compute; (2) solving the expectation of Equation 2.1 is as hard as directly checking for output similarity between f and f^*, i.e. output alignment; and (3) quantities related to f^* are intractable because f^* is unknown to us. Throughout Section 2.2 to Section 2.4, we discuss practical solutions to analyze (ρ, λ)-feature alignment using *explainability* tools, object bounding boxes and a new set of metrics which we will later term as *locality* scores. We also summarize this overview in Table 2.1.

2.2 Feature Attribution

Localizing $\mathcal{I}_f^\rho(x)$, i.e. the ρ-important features of x that contribute to the prediction $f(x)$, is the first step towards evaluating perceptional alignment. Despite the fact that one can iterate over samples from the noise distribution N to directly compute $\mathcal{I}_f^\rho(x)$ in Definition 1, it is not clear how many samples are enough for us to represent the distribution of interest well. As a result, Definition 1 may bring a great computation overhead, provided that we must iterate over noise sampled from N for each input dimension and repeat the same procedure for every input.

Alleviating the computation overhead while capturing the important features used by the model is in line with the focus of deep learning *explainability*, which includes a family of techniques to help humans understand and assess the behavior of deep nets. Among explanation techniques, *feature attributions* assign a scalar value for each input feature for its importance towards a quantity of interest, e.g. the output logit of the top class (Binder et al., 2016; BOHNENBLUST et al., 1952; Erion et al., 2021; Fong & Vedaldi, 2017; Ghalebikesabi et al., 2021; Leino et al., 2018; Lundberg & Lee, 2017; Pan et al., 2021; Petsiuk et al., 2018; Ribeiro et al., 2016; Selvaraju et al., 2019; Shrikumar et al., 2017; Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017; Wang et al., 2020a, 2022) (we give a brief review of other explainability tools and examples in Appendix A). The return of feature attributions thus fits our purpose of localizing $\mathcal{I}_f^\rho(x)$ in the input. We provide a general definition of an attribution method in Definition 3.

Definition 3 (Feature Attribution) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m, f \in \mathcal{F}$, feature attribution is a mapping*

$A : \mathbb{R}^d \times [m] \times \mathcal{F} \to \mathbb{R}^d$ that takes an input x, the network f to explain and a class of interest y and returns a vector of the input's shape. The j-th element of $A(x, f, y)_j$ indicates the importance of x_j.

Throughout the thesis, when f and y are clear from the context, we simply write an attribution as $\mathsf{A}(x)$. Notice that the distribution of interest to specify the neighborhood for feature importance is often part of the construction of $\mathsf{A}(x)$, where a reference neighborhood could be chosen for well-justified reasons or just being empirically feasible. One example of $\mathsf{A}(x)$ with an axiomatically justified distribution of interest is Shapely Value (BOHNENBLUST et al., 1952), a construct from cooperative game theory. Despite the fact that Shapely Value does not scale up well to high-dimensional input and deep models, it is the unique approach that satisfies a set of axioms considered "golden rules" for an explanation.

In this thesis, we are particularly interested in gradient-based attributions that are axiomatically justified to be faithful to the model w.r.t a neighborhood of interest. Compared to Shapley Values, gradient-based attributions are much more resource-friendly and often implementation-invariant as long as the underlying model is differentiable. In Section 2.2.1, we discuss three popular gradient-based attributions, followed by Section 2.2.2 where we propose a new type of gradient-based attribution that better captures the network's decision in a *local* neighborhood. In Section 2.2.3, we justify why the selected attributions are faithful enough to accurately capture $\mathcal{I}_f(x)$, the important features used by the model, to realize the purpose of assessing the alignment.

2.2.1 Saliency Map and Integrated Gradient

Saliency Map (Simonyan et al., 2013) (Definition 4) is the weight of the first-order approximation to the network's output at the input of interest. Because the weight $\mathsf{A}_S(x)$ is often considered as the sensitivity of f to the input features, e.g. how much the output would change if a feature is modified with a tiny perturbation, the product of the Saliency Map and the value of the feature, i.e. $\mathsf{A}_S(x)_j \cdot x_j$, is often considered as the contribution of x_j under the Saliency measurement. In the rest of the thesis, we will use Input $\times$ Grad if we particularly refer to $\mathsf{A}_S(x)_j \cdot x_j$ instead of the gradient itself.

Definition 4 (Saliency Map (SM) (Simonyan et al., 2013)) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m$, an input x, and a class of interest y, the Saliency Map $A_S(x)$ equals to*

$$A_S(x) = \frac{\partial f_y(x)}{\partial x}.$$

The feature importance found by $\mathsf{A}_S(x)$ can be thought of as the green case in Figure 2.1; thus, it is a *hyper-local* explanation for the model's behavior at the input of interest x. As a result, it may not be able to quantify the feature's importance in a wider neighborhood. An alternative to Saliency Map

is Integrated Gradient (Sundararajan et al., 2017) (Definition 5), which on the other hand measures the feature's importance in a global view. Formally, $\mathsf{A}_{IG}(x)$ integrates the gradient of the model's output w.r.t the input for all points on a linear path from a baseline input b to the input x to explain.

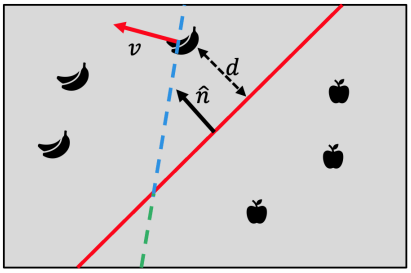
Definition 5 (Integrated Gradient (IG) (Sundararajan et al., 2017)) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m$, an input x, a class of interest y, and a baseline input b, Integrated Gradient $A_{IG}(x)$ equals to*

$$\begin{aligned} A_{IG}(x) &= \int_0^1 \frac{\partial f_y(r(t;x,b))}{\partial r} \frac{\partial r(t;x,b)}{\partial t} dt \\ &= (x-b) \cdot \int_0^1 \frac{\partial f_y(r(t;x,b))}{\partial r} dt \text{ where } r(t;x,b) = b + (x-b)t. \end{aligned}$$

To see why $\mathsf{A}_{IG}(x)$ is measuring the global geometry of the model's behavior, it is useful to think of the gradients of points on $b \to x$ as the first-order approximation to function, the integral of which averages a set of local geometry information obtained from these points. As a result, $\mathsf{A}_{IG}(x)_j$ is the contribution of the j-th feature when its value changes from b_j to x_j. Care must be taken that the choice of baseline b determines which linear path (and the corresponding global geometry) is of interest. Usually, b is set to a black image, i.e. a vector of all 0s, for image data because the changes from a black image to the current one resemble the process of shedding a light onto the input so the relevant features gradually show up when moving towards x. The same baseline may not work properly for other data types, e.g. graphs as exemplified by (Wang et al., 2023).

To summarize, the feature importance scores of the Integrated Gradient differ from Saliency Map by respecting a larger input neighborhood. One caveat for using the black image (i.e. $b = \mathbf{0}$) as the baseline is that the corresponding neighborhood varies from one instance to another and is sometimes sensitive to particular feature values. For instance, for input that is near the original point, Integrated Gradient captures a much local neighborhood compared to the point that is further away. That is, if $||x - b|| \to 0$, Integrated Gradients may converge towards Saliency Map. Moreover, for feature values close to 0 (i.e. $x_j \approx 0$), its contribution is often close to 0 as well because of the product between $x_j - b_j$ and the gradient integral in Definition 5. Thus, the black background of MNIST images usually receives small importance scores even though the model is possible to use noises in the background to classify the input.

In contrast to a hyper-local or a global neighborhood, this thesis is more interested in measuring the feature importance in a fairly local neighborhood around the point of interest and capturing the model's local behavior. In Section 2.2.2, we describe our new approach, Boundary-based Integrated Gradient, for realizing such purpose.



(a) (b) (c)

Figure 2.2: Different classifiers that partition the space into regions associated with `apple` or `banana`. (a) A linear classifier where $\hat{\mathbf{n}}$ is the only faithful explanation and $\mathbf{v}$ is not. (b) A deep network with ReLU activations. Solid lines correspond to decision boundaries while dashed lines correspond to facets of activation regions. (c) Saliency map of the target instance may be normal to the closest decision boundary (right) or normal to the prolongation of other local boundaries (left).

2.2.2 Boundary-based Integrated Gradient

Boundary-based Integrated Gradient (BIG) is an attribution method, the score of which manifests the importance of a feature within a *local* neighborhood. That is, we propose a way to automatically select a baseline input b for Integrated Gradient (instead of letting b to be determined by a human user), which leads the gradient integral to capture the nearby decision boundaries. These nearby decision boundaries, compared to ones that are further away, are important local geometrical information of the model's output in the input space. To connect a decision boundary and the local geometry of a model's output, we start with a simpler case of linear models, which contain exactly one boundary, and then generalize to piecewise-linear classifiers as they are embodied by deep ReLU networks.

Attribution and Decision Boundary in Linear Models. Consider a binary classifier $C(x) = \text{sign}(w^\top x)$ that predicts a label $\{-1, 1\}$ (ignoring "tie" cases where $C(x) = 0$, which can be broken arbitrarily). In its feature space, $C(x)$ is a hyperplane H that separates the input space into two open half-spaces S_1 and S_2 (Figure 2.2(a)). Accordingly, the normal vector $\hat{n}$ of the decision boundary is the only vector that faithfully explains the model's classification while other vectors, while they may describe directions that lead to positive changes in the model's output score, are not faithful in this sense (see $\mathbf{v}$ in Figure 2.2(a) for an example). In practice, to assign attributions for predictions made by C, Saliency Map and the integral part of Integrated Gradient return a vector characterized by $z = k_1\hat{n} + k_2$ (Ancona et al., 2018b), where $k_1 \neq 0$ and $k_2 \in \mathbb{R}$, regardless of the input x that is being explained. In other words, these methods all measure the importance of features by characterizing the model's decision boundary, and are equivalent up to the scale and position of $\hat{n}$.

Generalizing to Piecewise-Linear Boundaries. In the case of a piecewise-linear model, such as a ReLU network, the decision boundaries comprise a collection of hyperplane segments that partition the feature space, as in H_1, H_2 and H_3 in the example shown in Figure 2.2(b). Because the boundary no longer has a single well-defined normal, one intuitive way to extend the relationship between boundaries and attributions developed in the previous section is to capture the normal vector of the *closest* decision boundary to the input being explained. However, as we show in this section, the methods that succeeded in the case of linear models (i.e. Saliency Map and Integrated Gradient) may in fact fail to return such attributions in the more general case of piecewise-linear models, but local robustness often remedies this problem. We begin by reviewing key elements of the geometry of ReLU networks (Jordan et al., 2019a).

ReLU Activation Polytopes. For a neuron u in a ReLU network $f(x)$, we say that its status is `ON` if its pre-activation $u(x) \geq 0$, otherwise it is `OFF`. We can associate an *activation pattern* denoting the status of each neuron for any point x in the feature space, and a half-space A_u to the activation constraint $u(x) \geq 0$. Thus, for any point x the intersection of the half-spaces corresponding to its activation pattern defines a polytope P (see Figure 2.2(b)), and within P the network is a linear function such that

$$\forall x \in \mathbb{R}^d, x \in P \implies f(x) = w_P^\top x + b_p,$$

where the parameters w_P and b_P can be computed by differentiation (Fromherz et al., 2021a). Each facet of P (dashed lines in Figure 2.2(b)) corresponds to a boundary that "flips" the status of its corresponding neuron. Similar to activation constraints, decision boundaries are piecewise-linear because each decision boundary corresponds to a constraint $f_i(x) \geq f_j(x)$ for two classes i, j (Fromherz et al., 2021a; Jordan et al., 2019a).

Gradients Might Fail. Saliency maps, which we take to be simply the gradient of the model with respect to its input, can thus be seen as a way to project an input onto a decision boundary. That is, a Saliency Map $\mathsf{A}_A(x)$ is a vector that is normal to a nearby decision boundary segment. However, as others have noted, a Saliency Map is not always normal to any real boundary segment in the model's geometry (see the left plot of Figure 2.2(c)), because when the closest boundary segment is not within the activation polytope containing x, the Saliency Map will instead be normal to the linear extension of some other hyperplane segment (Fromherz et al., 2021a) In fact, iterative gradient descent typically outperforms the Fast Gradient Sign Method (Goodfellow et al., 2015a) as an attack demonstrates that this is often the case.

Boundary-based Attribution. In this section, we build on the insights of our analysis to present a set of novel attribution methods that explicitly incorporate the normal vectors of nearby boundary segments.

Using the normal vector of the closest decision boundary to explain a classifier naturally leads to Definition 6, which defines attributions directly from the normal of the closest decision boundary.

Definition 6 (Boundary-based Saliency Map (BSM)) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m$ and its integer prediction $F : \mathbb{R}^d \to [m]$, an input x, a class of interest y, we define Boundary-based Saliency Map $A_{BS}(x)$ as follows:*

$$A_{BS}(x) = \frac{\partial f_y(x_{adv})}{\partial x_{adv}},$$

where x_{adv} is the closest adversarial example to x,

so $F(x) \neq F(x_{adv})$ but $\forall x_m, ||x_m - x|| < ||x_{adv} - x|| \implies F(x) = F(x_m)$.

One limitation of using Definition 6 is obvious: this is still a *hyper-local* method as Saliency Map whereas our purpose is to find a *local* attribution method. Thus, our goal is to incorporate more nearby decision boundaries around x. The fact is that even in a small network, there will be numerous boundary segments in the vicinity of a relevant point. Taking inspiration from Integrated Gradients, Definition 7 proposes the Boundary-based Integrated Gradient (BIG) by aggregating the attributions along a line between the input and its closest boundary segment.

Definition 7 (Boundary-based Integrated Gradient(BIG)) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m$ and its integer prediction $F : \mathbb{R}^d \to [m]$, an input x, a class of interest y, we define Boundary-based Integrated Gradient $A_{BIG}(x)$ as follows:*

$$A_{BIG}(x) = (x - x_{adv}) \cdot \int_0^1 \frac{\partial f_y(r(t; x, b))}{\partial r} dt,$$

where $r(t; x, x_{adv}) = x_{adv} + (x - x_{adv})t$.

Here x_{adv} is the nearest adversarial example to x. That is,

$$F(x) \neq F(x_{adv}) \text{ and } \forall x_m . ||x_m - x_{adv}|| < ||x - x_{adv}|| \implies F(x_m) = F(x).$$

Geometric View of BIG. BIG explores a linear path from the boundary point to the target point. Because points on this path are likely to traverse different activation polytopes, the gradient of intermediate points used to compute g_{IG} are normals of linear extensions of their local boundaries. As the input gradient is identical within a polytope P_i, the aggregate computed by BIG sums each gradient w_i along the path and weights it by the length of the path segment intersecting with P_i. In other words, one may view IG as an exploration of the model's global geometry that aggregates all boundaries from a fixed reference point, whereas BIG explores the local geometry around x. In the former case, the global exploration may

Figure 2.3: Visualizations of geometrical interpretations of Saliency Map (SM), Boundary-based Saliency Map (BSM), Integrated Gradient (IG) and Boundary-based Integrated Gradient (BIG).

reflect boundaries that are not particularly relevant to the model's observed behavior at a point, whereas the locality of BIG may aggregate boundaries that are more closely related. This geometrical comparison is illustrated in Figure 2.3. Gradient computations are depicted as projecting the input onto a particular decision boundary. While SM projects to a nearby boundary (H_1), BSM projects to the nearest one (H_2). IG (the red dashed path) from a global baseline b (e.g. $b = \mathbf{0}$) aggregates boundaries in colorful shaded areas; BIG (the green dashed path) integrates from the point x_{adv} on the nearest boundary H_2 to x and therefore aggregates nearby boundaries, H_1 and H_2 in gray shaded areas.

Finding nearby boundaries. Finding the exact closest boundary segment is equivalent to the problem of certifying local robustness (Fromherz et al., 2021a; Jordan et al., 2019a; Kolter & Wong, 2018; Lee et al., 2020a; Leino et al., 2021c; Tjeng et al., 2019b; Weng et al., 2018b), which is NP-hard for piecewise-linear models (Sinha et al., 2020). To efficiently find an approximation of the closest boundary segment, we leverage and ensemble techniques for generating adversarial examples, i.e. PGD (Madry et al., 2018b), AutoPGD (Croce & Hein, 2020a) and CW (Carlini & Wagner, 2017a), and use the closest one found given a time budget. The details of our implementation will be discussed in Section 3.1 of this chapter and Section 3.3 of Chapter 3.

2.2.3 Attribution Faithfulness

Section 2.2.1 introduce Saliency Map (SM) and Integrated Gradient (IG) that one can employ to find the importance of an input feature w.r.t a *hyper-local* and *global* neighborhood, respectively. As a complementary to SM and IG, Section 2.2.2 introduces Boundary-based Integrated Gradient (BIG) to find the importance of an input feature w.r.t a *local* neighborhood. In this section, we justify that these three approaches are

both *faithful* to the network f in their corresponding input neighborhoods. To date, there are many ways to define faithfulness for attributions of a deep model. Recall our use of feature attribution is to locate $\mathcal{I}_f^\rho(x)$ (i.e. the ρ-important features to the model), we need attribution to have a reasonable sensitivity to the change of the input (within the corresponding neighborhood) and to the change of network parameters. We refer to these two properties as Distributional Faithfulness and Parameter Faithfulness, respectively.

Distributional Faithfulness. Attributions must be sensitive to the change of features, e.g. $x+\epsilon$, in the input but should only account for the output change, i.e. $f(x+\epsilon)-f(x)$, and no more than that. For this reason, Yeh et al. (Yeh et al., 2019) introduce the INFD score (Definition 8) to measure the deviation of an attribution method from precisely accounting for the output change of the model. Namely, a lower INFD score indicates that the underlying attribution is more faithful.

Definition 8 (INFD (Yeh et al., 2019)) *For a network $f : \mathbb{R}^d \to \mathbb{R}^m$, a distribution of interest $\mathcal{N}$, a feature attribution A that explains the output of class y, the INFD score for A is defined as*

$$INFD(A) = \mathbb{E}_{\epsilon\sim\mathcal{N}}\left[A(x)^\top\epsilon - (f_y(x+\epsilon) - f_y(x))\right]^2.$$

To put it simply, Definition 8 measures the goodness of using the attribution score as the weight to linearly approximate the function within the neighborhood defined by the distribution $\mathcal{N}$. Notice that Definition 8 measures the *difference of difference*, i.e. the difference between the change explained by the attribution score $\mathsf{A}(x)^\top\epsilon$ and the change of model's output $(f_y(x+\epsilon)-f_y(x)$, instead of the *difference between outputs*, e.g. $\mathsf{A}(x)^\top x - f_y(x)$, because we do not care if the attribution accounts for the bias, if any, of the model. For example, if the target model to explain is $f(x) = \sum_i w_i x_i + w_{bias}$, an attribution is sufficiently faithful to account for the change of output caused by $\sum_i w_i x_i$ and ignore w_{bias}. Thus, $f_y(x+\epsilon) - f_y(x)$ removes the bias term if any. A limitation of Definition 8 is that it assumes that the attribution scores must be linearly combined with the feature, one more generic form of which can be in Wang et al. (Wang et al., 2023).

- The Faithfulness of SM. As shown by Yeh et al. (Yeh et al., 2019), SM is the most faithful method, i.e. the one that minimizes INFD score, w.r.t to the delta distribution at x. Namely, when the width of $\mathcal{N}$ goes to 0, the INFD also goes to 0 if $\mathsf{A}(x) = \mathsf{A}_S(x)$.
- The Faithfulness of IG and BIG. Because IG is a *complete* attribution (Sundararajan et al., 2017), i.e. $\sum_j \mathsf{A}_{IG}(x)_j = f_y(x) - f_y(b)$, it is obvious that (the integral part of) IG minimizes INFD if the distribution $\mathcal{N}$ is a delta distribution at the baseline b. However, we are more interested in the faithfulness of IG w.r.t a uniform distribution over a linear line between the baseline and the input

Attribution	Neighborhood	Faithfulness	User-defined Baseline	Using Adv. Attack
Saliency Map	hyper-local	optimal to INFD	✗	✗
Integrated Gradient	global	completeness	✓	✗
Boundary-based Integrated Gradient	local	completeness	✗	✓

Table 2.2: Comparing Saliency Map, Integrated Gradient and Boundary-based Integrated Gradient.

because this is the same distribution that the integral part of IG aggregates over. In theory, one of the way to minimize INFD w.r.t the uniform distribution over the linear path $\mathcal{U}$ is to: 1) use every point $u \sim \mathcal{U}$ as a baseline and compute the integral of gradients from u to x; and 2) aggregate over these integrals (Yeh et al., 2019). Therefore, IG is suboptimal for minimizing INFD over the linear path because it only minimizes a proxy of the objective, i.e. INFD score over a deterministic perturbation with the starting point of the path. In practice, the implementation of the optimal attribution is infeasible due to a great computation overhead so we still use IG instead. Our analysis of IG should directly applies to BIG because they are only different on the choice of the baseline.

Parameter Faithfulness. Attributions must be sensitive to the parameters of the network because a degenerated model is not supposed to share the same explanation with a well-generalized one. Adebayo et al. (2018) propose a sanity check to determine the parameter faithfulness of an attribution. That is, we repeat the computation of feature attribution after parameters at each layer are replaced with random numbers from top to bottom. A faithful attribution is expected to have different results, measured by a similarity metric, e.g. Rank Order Correlations between the absolute values of attribution scores (Adebayo et al., 2018), compared to the attribution of a model without randomization. Since Adebayo et al. (2018) have shown that SM and IG have successfully passed this sanity check so we here only include the experiment results for BIG. We use a pretrained ResNet50 (He et al., 2015) on ImageNet that is publicly available on Pytorch and gradually randomize the weights for 5 layers at each time from top to bottom until we have a fully random model. Notice that to ensure the randomization does not produce `NaN` output we ensure the randomized weights have the same norm. The result for BIG is shown in Figure 2.4. We consider BIG passes the sanity check as the result is similar compared with the top row of Figure 4 in Adebayo et al. (2018).

2.2.4 Summary and Discussion

From Section 2.2.1 to Section 2.2.3, we introduce three types feature attributions: Saliency Map, Integrated Gradient, and Boundary-based Integrated Gradient for localizing important features to the network's prediction. Their importance scores are measured w.r.t a hyper-local, global and local neighborhoods,

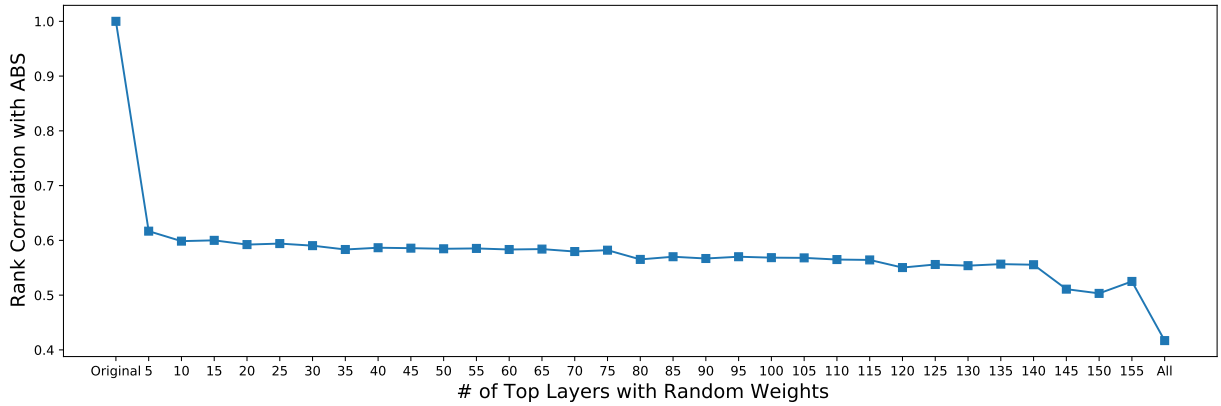


Figure 2.4: The rank order correlations of the absolute values of BIGs against the number of layers (counting from top to bottom) where trainable weights are replaced with random matrices.

respectively. To ensure these attributions truly manifest the model's behaviors in the corresponding neighborhood, we demonstrate their distributional and parameter faithfulness in Section 2.2.3. Comparisons between these attributions are also summarized in Table 2.2.

Besides the choices in this thesis, there are other candidate attributions with similar reference neighborhoods. The following paragraphs briefly discuss the related works and the reasons why we use Saliency Map, Integrated Gradient, and Boundary-based Integrated Gradients over others.

Other hyper-local Attributions. Candidates that also find hyper-local important features include Guided Backpropagation (GB) (Springenberg et al., 2014), Layer-wise Relevance Propagation (LRP) (**?**), methods based on Class Activation Map(CAM) (Selvaraju et al., 2019; Wang et al., 2020a; Zhou et al., 2016). Compared to Saliency Map, GB only backpropagates positive gradients from the output to the input by filtering out negative gradients at each activation layer. Adebayo et al. (Adebayo et al., 2018) have shown with empirical results that GB is not a faithful method because of its failure to pass the sanity check discussed in Section 2.2.3; therefore, we do not use GB in this thesis. Similarly, LRP also modifies the gradient backpropagation with different rules and some of these rules also fail to be faithful to the model (Adebayo et al., 2018). On the other hand, the implementation of LRP rules is not invariant to the network architecture so one needs to implement LRP for each type of layer in the network. Lastly, CAM-based methods aggregate channels of the internal activation, e.g. from the last convolution layer, and up-sample to the input size for identifying important features. Because there is no faithfulness justification for CAM-based methods, we do not consider this set of methods in this paper.

Other Global Attributions. Candidates that also explore globally important features include DeepLIFT (Shrikumar et al., 2017), Occlusion-N (Ancona et al., 2018a), and a set of baselines for Integrated Gradient (Erion

et al., 2021; Yeh et al., 2019). Similar to LRP, DeepLIFT (Shrikumar et al., 2017) also modifies the gradient backpropagation but it ensures the same *completeness* property as Integrated Gradient. As a result, DeepLIFT is not implementation-invariant for network architecture and thus is less flexible compared to Integrated Gradient. Occlusion-N is a global attribution because it can be thought as measuring the contribution of the chosen N features from baseline values to the current ones. Occlusion-N (Ancona et al., 2018a) is computationally expensive because of a good number of inferences runs is required when N is small, e.g. N=1. On the other hand, if N » 1, we must assign the same importance scores for N features, which may over-simplify the way the model processes the input and leads to an unfaithful attribution map. Lastly, besides choosing one baseline, recent works also propose to use multiple baselines and average Integrated Gradients computes over these baselines (Erion et al., 2021; Yeh et al., 2019). As is briefly discussed in Section 2.2.3 and detailed by Yeh et al. (Yeh et al., 2019), aggregations of Integrated Gradients can minimize INFD scores w.r.t some distribution of interest but it often comes with a significant computation overhead especially for large models. As a result, this thesis only uses Integrated Gradient.

Other Local Attributions. A similar line of work to Boundary-based Integrated Gradient is AGI (Pan et al., 2021). AGI is motivated to find a non-linear path that is linear in the representation space, instead of a linear line in the input space (i.e. the path integral used in Integrated Gradient and Boundar-based Integrated Gradient). AGI uses Projected Gradient Descend (PGD) (Madry et al., 2018b), an iterative approach to search for another input that has a different prediction as the underlying input. After PGD stops, AGI aggregates gradients on the non-linear path generated by the PGD search. The path of PGD attack can be twisted, circular, and even broken lines either due to the projection or the overlook of higher-order terms in the derivative for efficiency reasons. We believe such non-linearity in the path integral might not be necessary. Besides line integral, aggregating the gradients over a set of points sampled from a Gaussian distribution is another popular way of capturing the local geometry of the model, which is known as Smooth Gradient (Smilkov et al., 2017). In Section 3.3 of Chapter 3, we provide greater details on comparing Smooth Gradient with Saliency Map and Boundary-based Integrated Gradient and conclude that Smooth Gradient is equivalent to computing Saliency Map in a model with smoothed decision boundary compared to the original model. As a result, we do not use Smooth Gradient to capture the behavior of the current (i.e. non-smoothed) model.

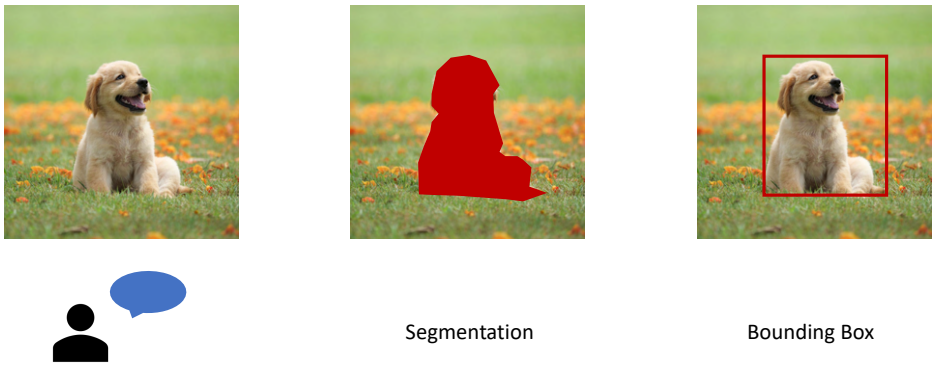


Figure 2.5: Important features can be found by human oracles (left) and approximated by a segmentation output (middle) or a bounding box (right).

2.3 Capturing Truly Important Features with Bounding Box

In Section 2.2, we describe our use of feature attributions for finding a set of important features as a faithful way to seek for $\mathcal{I}_f^\rho(x)$, i.e. the important features to DNNs in an empirical way. In this brief section, we discuss an empirical way for localizing $\mathcal{I}_{f^*}^\rho(x)$, i.e. the important features to humans, using human-labeled bounding box information.

The ideal way of locating the important features is to have an oracle, e.g. a human, to give locations or even continuous values for the importance of each pixel. To date, there is a large body of work that conducts a user survey for the results of attributions. However, the only useful information from any human-based study to our purpose is the whereabouts of the object in the image that is associated with the label, which is often provided in many standard vision datasets, e.g. ImageNet (Deng et al., 2009), as a segmentation or a bounding box shown in Figure 2.5. Despite the limitation that segmented regions and bounding boxes only indicate where this pixel is relevant or not, they are still human-labeled and thus faithful to the true mapping f^* of the data distribution.

For localizing the aligned set of features $\mathcal{I}_{f^*}^\rho(x)$, we use the bounding box instead of the segmentation results in this chapter. The advantage of object segmentation is that it contains fewer pixels from unrelated pixels so it provides a more precise way to group important pixels compared to a bounding box, e.g. Figure 2.5. However, this thesis is more interested in the change of feature alignment as a result of training instead of the actual values. Besides, bounding boxes are easier to assess and presented together with classification labels in ImageNet. Because of these two reasons, we decide to use bounding boxes to localize $\mathcal{I}_{f^*}^\rho(x)$ for ImageNet images in the rest of the thesis.

2.4 Locality

With the assistance of feature attributions discussed in Section 2.2 and the use groundtruth bounding boxes discussed in Section 2.3, this section proposes a new metric, *locality*, as a practical way of assessing (ρ, λ)-feature alignment.

For feature attributions introduced in this thesis, i.e. Saliency Map (SM), Integrated Gradient (IG) and Boundary-Based Integrated (BIG), we use their attribution scores as a proxy for ρ-important features. Recall Definition 1 where a feature x_j is ρ-important if

$$\mathbb{E}_{\epsilon\sim\mathcal{N}}[(x_j+\epsilon)\cdot f_y(x_1,\cdots,x_j+\epsilon,\cdots,x_d)] \geq \rho. \tag{2.2}$$

By using attribution scores as proxy for ρ-importance, we say a feature feature x_j is ρ-important w.r.t A_* if

$$\mathsf{A}_*(x)_j \geq \rho.$$

where $*$ can be S, IG or BIG. Notice that the distribution of interest, $\mathcal{N}$, in Equation 2.2 corresponds to the different neighborhoods of interests used in SM, IG, and BIG. That is, $\mathcal{N}$ corresponds to a delta distribution at x for using SM $\mathsf{A}_S(x)$ and uniform distributions over lines for using Integrated Gradient $\mathsf{A}_{IG}(x)$ and Boundary-based Integrated Gradient $\mathsf{A}_{BIG}(x)$. For the choice of ρ, we simply set $\rho = 0$[1] to include all features with positive correlations to the output. When $\rho = 0$ is clear from the context, we directly write $\mathcal{I}_f(x)$ and formally use

$$\mathcal{I}_f(x)_j = \mathsf{A}_*(x)_j \cdot \mathbb{I}[\mathsf{A}_*(x)_j > 0] \quad (* \text{ can be } S, IG \text{ or } BIG).$$

Similarly, we translate the bounding box coordinates into a set of pixels in that bounding box, which is considered to be input used by the true function f^* employed by humans. In this case, we directly write $\mathcal{I}_{f^*}(x)$ as

$$\mathcal{I}_{f^*}(x)_j = \mathbb{I}[j \text{ in the bounding box of } x].$$

2.4.1 Locality Metrics

Our use of feature attribution and the bounding box information results in a continuous and discrete approximations for the important features to the model $\mathcal{I}_f(x) \in \mathbb{R}^d_+$ and truly important features $\mathcal{I}_{f^*}(x) \in \{0,1\}^d$, respectively. To compare the similarity between $\mathcal{I}_f(x)$ and $\mathcal{I}_{f^*}(x)$, we define precision (Definition 9, also known as Energy Game in (Wang et al., 2020a)) and recall scores (Definition 10, also known as Positive

[1]Notice that one can use other thresholds for ρ to obtain scores similar to mAP. For our purpose of evaluating locality in this thesis, varying the threshold is perhaps not necessary so we will stick with setting $\rho = 0$.

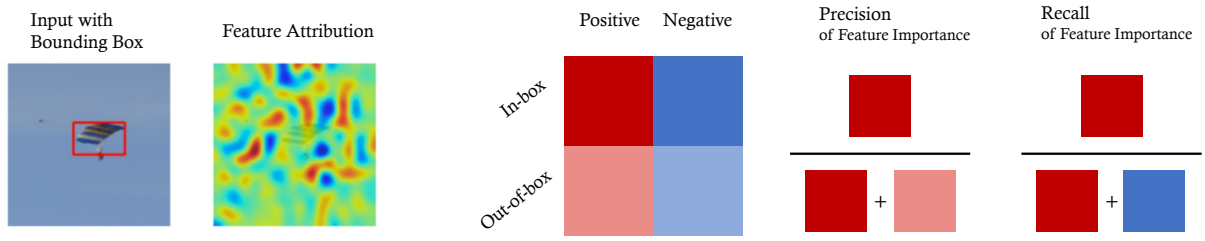


Figure 2.6: An illustration of locality with a precision-like and a recall-like similarity metrics, respectively.

Portion (Wang et al., 2022)) for feature importance. An illustration of these two metrics is included in Figure 2.6.

Definition 9 (Precision Locality) *For a network f and an input x, let $\mathcal{I}_f(x)$ be the important features to the model approximated by an attribution $A(x)$ and $\mathcal{I}_{f^*}(x)$ be the truly important features approximated by the bounding box of x. The precision Pre_f of feature importance to the network f equals to*

$$\text{Pre}_f(x) = \frac{\sum_j \mathcal{I}_f(x)_j \mathcal{I}_{f^*}(x)_j}{\sum_j \mathcal{I}_f(x)_j}.$$

Definition 10 (Recall Locality) *For a network f and an input x, let $\mathcal{I}_f(x)$ be the important features to the model approximated by an attribution $A(x)$ and $\mathcal{I}_{f^*}(x)$ be the truly important features approximated by the bounding box of x. The recall Rec_f of feature importance for the network f equals to*

$$\text{Rec}_f(x) = \frac{\sum_j \mathcal{I}_f(x)_j \mathcal{I}_{f^*}(x)_j}{\sum_j |A_*(x)_j| \mathcal{I}_{f^*}(x)_j}.$$

The nuance between our Definition 9 (and 10) and the conventional definitions of precision (and recall) in binary classification is that $\mathcal{I}_f(x)$ is a continuous vector while the output label from a binary classifier is discrete. We do not binaries $\mathcal{I}_f(x)$ to have discrete values because it loses the magnitudes of attributions. In the spirit of using the geometrical mean between the precision and recall, i.e. F1-locality, to balance two metrics as follows.

$$\text{F1}_f(x) = 2 \cdot \frac{\text{Pre}_f(x)\text{Rec}_f(x)}{\text{Pre}_f(x) + \text{Rec}_f(x)}. \tag{2.3}$$

2.5 Chapter Summary

In summary, this chapter provides a formal definition of *feature alignment* (Definition 2) through the lens of feature correlations. Despite the soundness of using Definition 2 to evaluate the alignment of a deep

model, it is practically infeasible without knowing the true mapping, employed by humans' perception system and brains, from the data to the label. Our approach to arrive at a practical evaluation metric, i.e. *locality*, for feature alignment includes the following three steps.

First, we use feature attribution as a tool to assign importance scores to each input features. In addition to Saliency Map (SM) and Integrate Gradient (IG), two attribution methods in the prior works, we propose a novel attribution tool — Boundary-based Integrated Gradient (BIG). While SM and IG faithfully return hyper-local and globally important features, BIG concentrates on a more local neighborhood and is also faithful due to the same guarantee as IG. Secondly, we identify the bounding box in standard vision tasks as a proxy for perceptibly important features because it contains location information about the object associated to the label. With attributions and bounding boxes, the last step proposes a set of metrics, i.e. *locality* scores, by adapting well-known precision, recall and F1 scores.

In the following Chapter 3, we use the tools developed in this chapter to evaluate the feature alignment of the recent state-of-the-art classifiers.

Chapter 3

Locality As A Result Of Robustness

Deep vision models have been growing deeper and changed a lot from the classic AlexNet Krizhevsky et al. (2012) and VGG Simonyan et al. (2013) architectures to the recent Vision Transformers (Dosovitskiy et al., 2020) that leads the Top-1 accuracy on ImageNet. It is interesting to ask the following research question.

Are more accurate vision models also more aligned on features?

Using attribution-based locality metrics, we give an empirical answer to the question above in Section 3.1. However, the title of this chapter probably has spoiled the result — we do not find more accurate models have better F1-locality compared to the less accurate ones, and sometimes can even be as low as models with random weights. A low locality of highly accurate models may indicate the limitation of relying on test accuracy for solving the issue of feature misalignment.

Compared to accuracy, this chapter demonstrates that *adversarial robustness*, i.e. the resistance of a network's inference against semantically meaningless perturbations to the input, as a more *necessary* condition to better alignment, provided that humans are robust models. For example, adding salt-and-pepper noise to an image of dog does not prevent our brain from seeing the dog. Similarly, replacing the word "story" with the word "plot" in the sentence "...the story looks horrible and..." does not change the fact the sentence contains *negative* sentiment. If a deep model indeed manages to leverage the same features in the input as humans do, i.e. in the spirit of locality, the same perturbations should not fool the model's inference either. However, the fact is that deep networks have been found to be vulnerable to tiny perturbations. For example, with imperceptible noise injected into an image of a cat, a (misaligned) deep model would change its prediction from CAT to DOG as depicted in Figure 3.1.

The rest of this chapter is therefore organized as follows. In Section 3.2, we give a formal definition of adversarial robustness and the corresponding adversary we study in this chapter. Next, we provide a

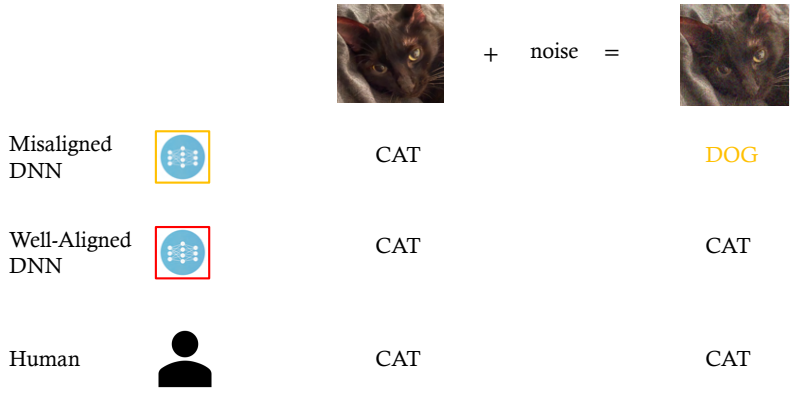


Figure 3.1: Humans and well-aligned DNNs are robust classifiers but misaligned DNNs are not.

set of theoretical and empirical results on showing adversarial robustness help to improve the locality of the model in Section 3.3. The promise of improving model robustness towards better locality motivates our study on training more robust neural networks. Towards this end, this thesis discusses two novel approaches that update the state-of-the-art robustness guarantee for deep vision models under empirical attacks (Section 4) and formal verification (Section 5), respectively.

3.1 Accuracy and Locality

The quality of a classification model is often measured by its accuracy on a test (or validation) set that is not seen by the model at the train time. The classes of test images are labeled by humans so accuracy is a proxy indicator of output alignment between the model and humans. However as we have pointed out test sets are often incomplete as we are unable to exhaust all samples from the true data distribution; therefore, we instead evaluate the feature alignment, as quantified by locality in this thesis.

In this section, we evaluate locality on classifying a subset of ImageNet, i.e. ImageNette (Howard), with a set of pre-trained models with different Top-1 accuracies. Our goal is to check whether the more accurate model indeed aligns better with our humans' perception. We first describe our experiment setup.

3.1.1 Experiment Setup

Models. We evaluate locality on the following pretrained models on ImageNette and note their Top-1 accuracy (measured on the full test set of ImageNet) in the parenthesis: SqueezeNet (58.1%) (Iandola et al., 2016); VGG16 (Simonyan & Zisserman, 2015)(71.6%); ResNet50 (He et al., 2015) (76.1%); ViT-B32 (Dosovitskiy et al., 2020) (75.9%); ViT-B16 (Dosovitskiy et al., 2020) (81.1%). SqueezeNet and VGG nets

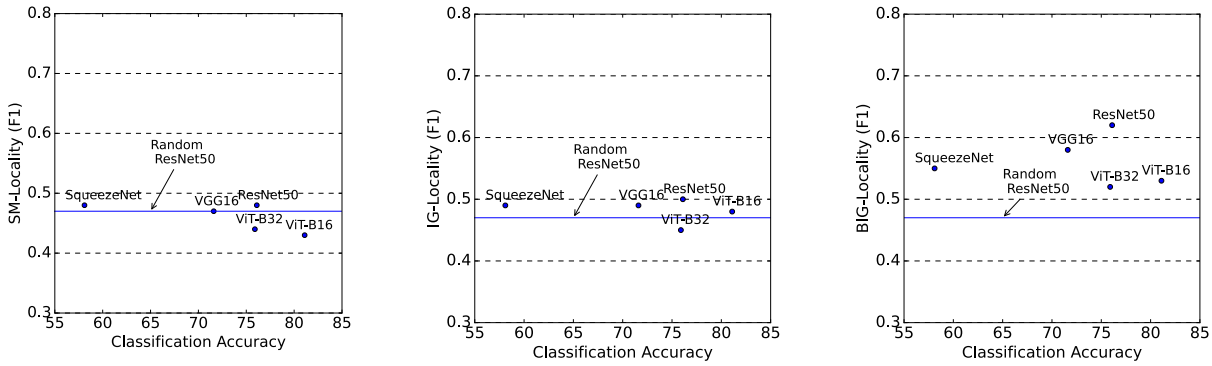


Figure 3.2: Plots of Locality v.s. Top-1 Accuracy for several pre-trained models. We measure F1-scores of locality (Equation 2.3) with Saliency Map (left), Integrated Gradient (middle) and Boundary-based Integrated Gradient (right). Results of averaged over 1000 images.

are classic architectures for vision classifications, while ResNets and ViTs are the modern architectures that provide state-of-the-art performances on ImageNet. Moreover, ViT differs from other architectures by replacing all convolution layers with attention layers so our conclusion does not just hold for CNNs. For a baseline comparison, we also use a ResNet50 with random weights.

Bounding Box. We use 1000 images from the test set of ImageNette with a bounding box area less than 80% the full area of the image. The selection is to avoid using well-cropped samples, e.g. the object is in the entire image and the bounding box is a bad approximation to the semantically meaningful features.

Attributions and Metrics. We use Saliency Map (SM), Integrated Gradient with a zero baseline (IG) and Boundary-based Integrated Gradient (BIG) to measure the important features to the model. We use 20 points to aggregate the integrals in IG and BIG. For the boundary search in BIG, we use a combination of PGD and CW attacks (see Appendix B for implementation details). We plot F1 for each model with every attribution in Figure 3.2 and leave results on Pre and Rec in Appendix B.

3.1.2 Results

On the x-axis of Figure 3.2 we label the Top-1 accuracy of each model reported on the test set of ImageNet, while the y-axis of report the F1 locality score measured with Saliency Map, Integrated Gradient and Boundar-based Integrated Gradient, respectively in each plot. For baseline comparison, we also repeat the same experiment with a random ResNet50. Ideally, one would expect F1 locality to go up as the accuracy increases. In practice, as the accuracy of the network goes up from the classic SqueezeNet to the recent Vision Transformer (ViT), their F1 locality scores do not differ significantly from one another.

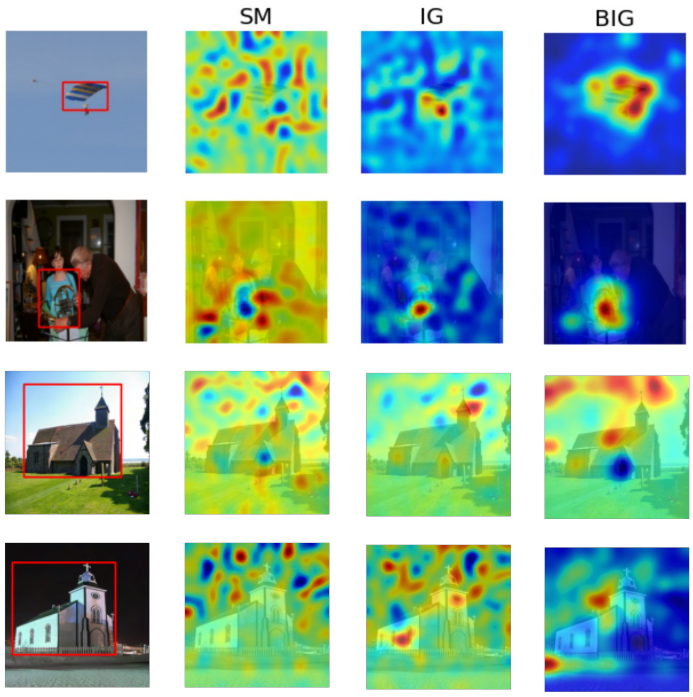


Figure 3.3: Plots of visualizations of Saliency Map (SM), Integrated Gradient (IG) and Boundary-based Integrated Gradient (BIG) on pre-trained ResNet50. Attributions are visualized as heatmaps using Trulens (Leino et al., 2021a) so positive scores are in red and negative scores are in blue.

The observation is not too surprising. As prior works have shown that deep networks can generalize to the given test set with spurious and high-frequency features (Ilyas et al., 2019; Wang et al., 2019), which are often imperceptible to humans as well. Compared to these prior works, our results directly show that even the state-of-the-art classifiers could have relied on features that are improper for the task at hand and thus should not be used in deployment. Again we want to emphasize that this is not a pitfall of attributions or the metric but instead a problem of the underlying model. As we will show in the next chapter when the quality of the model, e.g. robustness, is improved, the locality will be improved accordingly.

The second observation from Figure 3.2 is that F1 locality on pre-trained models are not significantly distinguishable from random models, except on the locality measured with BIG. This observation is not too surprising as SM and IG often look semantically meaningless on standard models as exemplified in Figure 3.3. Visualizations on BIG look sharper than SM and BIG but still may highlight irrelevant features. This observation is a demonstration that even though in general we do not believe improved accuracy leads to improved locality, these models are still expected to have learned partially useful features compared to a random model, which, however, can not be shown by SM and IG.

3.2 Adversarial Robustness

An adversary considered for a deep model is usually a third-party which aims to cause the malfunction to the model and maximize the adversary's utility. For example, directly injecting well-crafted noise to the input (Carlini & Wagner, 2017b; Croce & Hein, 2020a; Goodfellow et al., 2015b; Madry et al., 2018b) or creating physical objects to fool the camera of a vision model (Du et al., 2022; Kurakin et al., 2016; Lee & Kolter, 2019; Sharif et al., 2016). The type of attack considered in this thesis is the former, i.e. after an image is taken by the camera. Because the adversary's goal is to fool the deep model and keep the input perceptionally benign to humans (otherwise it is not reasonable to require the model to be consistent on outputs when humans also disagree), an adversary is often given a limited budget, i.e. a maximum amount of perturbation. In this thesis, we consider an adversarial perturbation is norm-bounded with a small radius so the perturbation is often imperceptible to humans.

Recall that in Chapter 2 we use $f : \mathbb{R}^d \to \mathbb{R}^m$ to denote a neural network that takes an input $x \in \mathbb{R}^d$ and outputs a logit vector for m classes. We use the uppercase $F : \mathbb{R}^d \to [m]$ to denote the integer prediction of the network such that $F(x) = \arg\max_j f_j(x)$. In addtion, we use $||\cdot||_p, 1 \leq p \leq \infty$, to denote the ℓ_p norm of a vector. A machine learning developer who aims to defend against a norm-bounded adversary therefore considers enforcing *Local Robustness* (Definition 11) to the model. Frequent choices of p in Definition 11 are $1, 2$ and ∞ and this thesis will focus on the 2 and ∞ cases.

Definition 11 (ϵ-Local Robustness) *A network $F(x) = \arg\max_j f_j(x)$ is ϵ-Local Robustness at x w.r.t norm, $||\cdot||_p$, if*

$$\forall x' \in \mathbb{R}^d, ||x' - x||_p \leq \epsilon \implies F(x') = F(x).$$

Evaluating Local Robustness. To check for the percentage of data on which the model is locally robust, we calculate the robust accuracy on a dataset $D^n = \{(x_i, y_i)\}_{i=1}^n$ as follows

$$RobustAcc.(D^n) = \frac{1}{n} \sum_i \max_{||\delta||_p \leq \epsilon} \mathbb{I}[F(x_i + \delta) = y_i]. \tag{3.1}$$

The maximization in Equation 3.1 is either approximated with an empirical attack, e.g. Projected Gradient Descend (PGD (Madry et al., 2018b)) or relaxed and upper-bounded by a certification approach, e.g. using the Lipschitz Constant of the function (Leino et al., 2021c). We refer to the robust accuracy found by an empirical attack as *Empirical Robust Accuracy* (ERA) and the one returned by a certification process as *Verifiable Robust Accuracy* (VRA) in the rest of the thesis.

3.3 Robustness and Locality

In this section, we run experiments to provide empirical evidence about our hypothesis – robust models are better aligned with our perception as they are not easily fooled by imperceptible noises.

3.3.1 Improved Locality with Robustness

Firstly, we describe the setup of our experiments.

Models. In Chapter 2, we evaluate the locality on pretrained and non-robust SqueezeNet, VGG16, ResNet50, ViT-B32 and ViT-B16. For apple-to-apple comparisons, we use their robust versions[1] that are trained under ℓ_∞-norm-bounded adversaries with a raidus of $\epsilon = 4/255$ on ImageNet. Unfortunately we are unable to find a robust SqueezeNet so in this section we only use the rest four models. The ERA of these models are evaluated against AutoAttack (Croce & Hein, 2020b), a state-of-the-art adversarial attack for empirically evaluating local robustness. The ERAs for these models are reported as VGG16 (25.92%), ResNet50 (34.90%), ViT-B32 (37.38%) and ViT-B16 (43.04%) by Mao et al. (2022). Notice that the non-robust version of these models have almost 0% ERA under attack. For baseline comparison, we continuously use a ResNet50 with random weights.

Bounding Box. We use 1000 images from the test set of ImageNette (Howard), a subset of ImageNet, with a bounding box area less than 80% of the full area of the image. The selection is to avoid using well-cropped samples, e.g. the object is in the entire image and the bounding box is a bad approximation to the semantically meaningful features.

Attributions and Metrics. We use Saliency Map (SM), Integrated Gradient with a zero baseline (IG) and Boundary-based Integrated Gradient (BIG) to measure the important features to the model. We use 20 points to aggregate the integrals in IG and BIG. For the boundary search in BIG, we use a combination of PGD and CW attacks (see Appendix B for implementation details). We plot F1 for each model with every attributions in Figure 3.4 (and leave Pre and Rec in Appendix B).

Secondly, we are ready to discuss our results shown in Figure 3.4, in which plots of F1 locality are sorted from left to right by the corresponding attribution methods: Saliency Map (SM, on the left), Integrated Gradient (IG, on the middle) and Boundary-based Integrated Gradient (BIG, on the right). The x-axis denotes the ERA of each robust model and the y-axis is the locality measured with F1-score (Equation 2.3). Scatter points in green are the results for robust models. To compare with the non-robust counterparts of

[1] weights are publicly available from Mao et al. (2022).

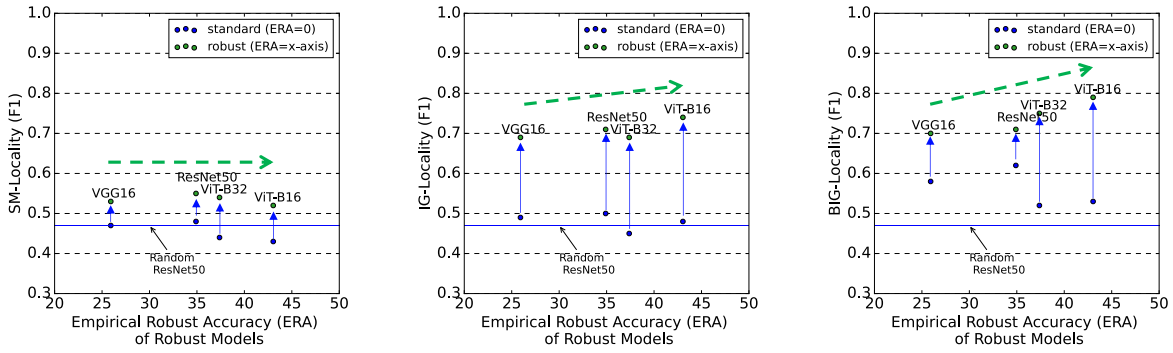


Figure 3.4: Plots of Locality v.s. Empirical Robust Accuracy (ERA) for robust models under ℓ_∞-norm-bounded adversaries with a radius of $4/255$. We measure the F1-score of locality (Equation 2.3) with Saliency Map (left), Integrated Gradient (middle) and Boundary-based Integrated Gradient (right). Results are averaged over 1000 images. Blue scatters are F1-scores measured on the non-robust counterparts, i.e. reusing results from Figure 3.2, for reference. The green arrays highlight the trend of locality against ERA.

the underlying models, we also add the scatter points from Figure 3.2 and color them in blue in the plots. We use the vertical blue arrows to highlight the improvement of locality on each model. The green arrow highlights how F1-score changes following the improvement of ERA across robust models. Now we are ready to discuss our observations.

Finding I: Robust models have higher locality than non-robust ones. In every plot, any green scatter, i.e. any robust model, point has better locality compared to all blue scatter, i.e. all non-robust models. In particular, we find the improvement of locality is most significant when the locality score is measured with IG. This observation indicates that the alignment between the globally important features and truly important features from humans' perspective benefit most from the improved robustness. On the other hand, the point-wise important features are benefited least.

Finding II: Improved Robustness Leads to Improved Locality. Using the green arrows, we see that the locality score measured with IG and BIG is improved as the ERA is improved. In particular, the improvement of locality is more obvious when measured with BIG. However, the locality measured with SM does not improve much as the model becomes more robust. This observation indicates that point-wise important features are less sensitive to the change of ERA, compared to globally and locally important features. Taking Finding II together with I, point-wise important features are influenced by the robustness of the model but such influence is limited. The conclusion is not surprising, as humans ourselves are very insensitive to infinitesimal changes made to small pixels in an 224×224 image and there is almost no such pixel, if added with infinitesimal noises, that would instantly change our response to the label of the input.

In summary, Figure 3.4 provides empirical evidence that robust models have higher locality compared to non-robust ones, and the higher the robustness the better the locality.

3.3.2 Attributions Better Capturing Boundaries in Robust Models

In Figure 3.2, we find that BIG has higher locality scores on F1-score compared to SM and IG, the gap of which becomes smaller in robust models (see Figure 3.4) than them in the standard (i.e. non-robust) models. It suggests that importance scores measured on a point-wise and global neighborhood are more similar to those measured in a local neighborhood in the robust models. In Figure 2.2 we have shown that gradients on the input of interest are probably very dissimilar to the normal vectors of the nearby decision boundaries because of the non-linearity of the neural networks. However, the observation of the small gap of locality between IG and BIG in robust models in Figure 3.4 seems to be a counter-example to the motivation of Figure 2.2. Consequently, we have to hypothesize that in the robust model, the local geometry of the model' output is more smooth, which therefore is much similar to a linear model compared to that in a non-robust model. This hypothesis should explain why IG is similar to BIG in robust networks.

To verify our hypothesis between robustness and the similarity of attributions, we run an empirical experiment in this section. BIG is a boundary counterpart of IG. Similarly, we have defined Boundary-based Saliency Map (BSM, Definition 6), a boundary counterpart for SM. The reason why we do not directly measure the similarity between SM and BIG is that by definition BG aggregates multiple normal vectors of decision boundaries while SM should only correspond to one of them. What we really want to show is that the gradient on the input (i.e. SM) is similar to the normal vector of the nearest decision boundary (i.e. BSM) and the average of gradients (i.e. IG) is also similar to the average of normal vectors of nearby decision boundaries (i.e. BIG). Therefore, SM-BSM and IG-BIG pairs are better fits here. We use ℓ_2 distance as a similarity measurement.

For the dataset and models used in this section, we use one standard ResNet50 and one robust ResNet50 with $\epsilon = 0.3$ in the training under an ℓ_2-norm-bounded adversary on CIFAR-10 (Krizhevsky et al.) dataset. On ImageNet, we use one standard ResNet50 and three robust ResNet50s trainind with $\epsilon = 3.0(\ell_2), \epsilon = 4/255(\ell_\infty)$ and $\epsilon = 8/255(\ell_\infty)$. These models are chosen over others because they are publicly available[2]. Results are shown in Table 3.1.

In Table 3.1, we find SM-BSM and IG-BIG are obviously smaller in the robust models than those in the standard ones. because for ℓ_∞ case we have two values of ϵ, we see that a higher ϵ, which is more robust, indicates a lower ℓ_2 distance between attributions. Particularly, using ℓ_2 norm and setting $\epsilon = 3.0$

[2] https://github.com/MadryLab/robustness (Engstrom et al., 2019).

CIFAR10	standard		$\ell_2\|0.5$	
SM-BSM.	59.96		1.23	
IG-BIG	31.22		2.73	
ImageNet	standard	$\ell_2\|3.0$	$\ell_\infty\|\frac{4}{255}$	$\ell_\infty\|\frac{8}{255}$
SM-BSM	8.48	0.41	2.25	1.61
IG-BIG	17.07	0.69	1.74	1.45

Table 3.1: ℓ_2 differences of SM-BSM and IG-BIG for standard and robust models. The heading of each column reports the respective training epsilon and the corresponding ℓ_p norm constraint

are most effective on ImageNet to have small SM-BSM and IG-BIG distances, compared to ℓ_∞ norm bound. One possible explanation is that the ℓ_2 space is special because training with ℓ_∞ bound may encourage the gradient to be more Lipschitz in ℓ_1 because of the duality between the Lipschitzness and the gradient norm, whereas ℓ_2 is its own dual.

3.3.3 Theoretical Justifications for Attribution Similarity

In this section, we provide two theoretical justifications (Proposition 1 and Theorem 1) on why attributions look similar to their boundary-counterparts i.e. those normal to the nearby decision boundaries.

Lipschizt Gradients. Recent work has proposed using the Lipschitz continuity of an attribution method to characterize the difference between the attributions of an input x and its neighbors within a ℓ_p ball neighborhood (Definition 12) (Wang et al., 2020c). This naturally leads to Proposition 1, which states that the difference between the saliency map at input and the correct normal to the closest boundary segment is bounded by the distance to that segment.

Definition 12 (Attribution Robustness) *An attribution method $A(x)$ is (λ, ϵ)-locally robust at the evaluated point x if*

$$\forall x', \|A(x') - A(x)\|_p \leq \epsilon \implies \|A(x') - A(x)\|_p \leq \lambda \|x' - x\|_p.$$

Proposition 1 *Suppose that f has a (λ, ϵ)-robust Saliency Map A_S at x, x' is the closest point on the closest decision boundary segment to x and $\|x' - x\|_p \leq \epsilon$, and that n is the normal vector of that boundary segment. Then $||n - A_S(x)|| \leq \lambda \|x - x'\|$.*

Proposition 1 therefore provides the following insight: for networks that admit robust attributions (Chen et al., 2019; Wang et al., 2020c), the Saliency Map is a good approximation to the boundary vector. As prior work has demonstrated the close correspondence between robust prediction and robust attributions (Cha-

lasani et al., 2020; Wang et al., 2020c), this in turn suggests that explanations on robust models will more closely resemble boundary normals.

Gradients in Randomized Smoothing. Proposition 1 abstracts a robust model with Lispchizt gradients after robust training. Another way to obtain a robust model is to use post-processing techniques like randomized smoothing (Cohen et al., 2019b) (Definition 13).

Definition 13 (Randomized Smoothing Cohen et al. (2019b)) *For a model $F : \mathbb{R}^d \to [m]$, its randomized smoothing model F_σ is*

$$F_\sigma(x) = \arg\max_c \{Pr[F(x+\epsilon) = c]\},$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I}\sigma^2)$ and $\mathcal{N}$ is a Gaussian distribution.

The robustness guarantee randomized smoothing provides is provable with a failure probability τ, which is referred to that the local robustness is guaranteed with a radius ϵ as a function of the model's output and the standard deviation σ of the used noise distribution in Equation 3.2. The failure probability τ, i.e. the probability the guarantee holds but there is actually an undetected adversarial example, decreases when it uses more samples to empirically approximate the probability function in Definition 13.

In particular, we prove the following Theorem 1, showing that for a single-layer network, the Saliency Map of the smoothed model is close to the Saliency Map of the original model up to the standard deviation of the noise used to smooth the model.

Theorem 1 *Let $m(x) = ReLU(Wx)$ be a one-layer network and its smoothed counterpart, $m_\sigma(x)$, introduced in Definition 13. Let $A_S^\sigma(x)$ be the Saliency Map for $m_\sigma(x)$. Given two points $x, x' \in \mathbb{R}^d$ such that $m_\sigma(x) = m_\sigma(x')$, we have the following statement holds:*

$$\|A_S^\sigma(x) - A_S^\sigma(x')\| \leq \lambda_a \tag{3.2}$$

where λ_σ is monotonically decreasing w.r.t σ.

Applying to Theorem 1 to the point of the decision boundary x_{adv}, we easily arrive at Corollary 1.

Corollary 1 *Let $m(x) = ReLU(Wx)$ be a one-layer network and its smoothed counterpart, $m_\sigma(x)$. Given a point x, its closest neighbor x_{adv} on the decision boundary and suppose $i = m_\sigma(x)$. If $A_S^\sigma(x)$, $A_S^\sigma(x_{adv})$ are the Saliency Map for the model m_σ w.r.t class i computed at x and x', then $||A_S^\sigma(x) - A_S^\sigma(x_{adv})|| \leq \lambda$ where λ is monotonically decreasing w.r.t σ.*

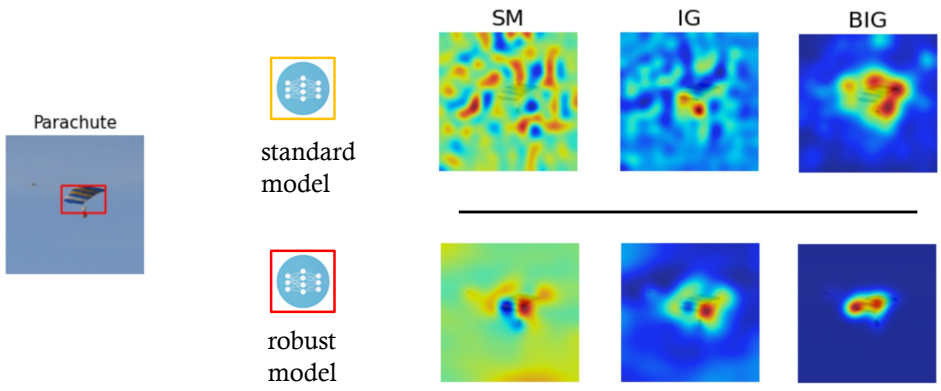


Figure 3.5: Visualizing Saliency Map (SM), Integrated Gradient (IG), and Boundary-based Integrated Gradient (BIG) on two images to compare standard and robust ResNet50 classifiers.

The Saliency Map of the closest neighbor x' on the decision boundary in Corollary 1 points to the same direction as the normal vector of the closest decision boundary we have discussed in the previous paragraph. Thus, this corollary suggests that when randomized smoothing is used, the normal vector of the closest decision boundary segment and the saliency map are similar, and this similarity increases with the smoothness of the model's boundaries. We think the analytical form for deeper networks exists but its expression might be unnecessarily complex due to that we need to recursively apply ReLU before computing the integral (i.e., the expectation). The analytical result above for one-layer networks and empirical validations for deeper nets in Table 3.1, if taken together, show that attributions and boundary-based attributions are more similar in a smoothed model.

3.4 Low-Quality Attributions Indicating Feature Misalignment

Robust models are much more perceptibly aligned with humans as evidenced by locality metrics. Indeed, if overlaying attributions on the same images with standard and robust ResNet50 classifiers (Figure 3.5), we see that attributions from the robust models are more semantically meaningful. The improvement on attributions is obvious to tell based on our common sense without locality metrics or bounding boxes.

The first take-away from our numerical evaluation on locality metrics and Figure 3.5 is that the perceived "low-quality" of gradient attributions, which has been justified to faithfully return important features to the model, is, in fact, demonstrating the feature misalignment of the underlying model. Thus, we should be skeptical about the (un)faithfulness of a feature explanation if it looks much more visually sharper than the faithful ones on models that do not always agree with humans, e.g. a non-robust classifier. It is hard to believe that a model that always fails on images that look benign to humans and only succeeds on a particular set of images actually takes the same set of features in the image for reasoning.

The second insight from this chapter is that robustness is an unspecified objective that we should put into the training if our goal is to obtain a classifier well-aligned with human-like perception. Issues seen on feature attributions should not lead to techniques that make attributions look better on misaligned models; instead, the way with a promising and high-quality explanation, in the end, is to have a robust model to begin with. In Chapter 4 and 5, we pursue this goal by discussing a set of techniques to make the underlying model adversarially robust against norm-bounded attacks.

Chapter 4

Training Empirically Robust Networks

Section 3.3 demonstrates the promise of improving the locality of deep models by improving their robustness against norm-bounded adversaries. In this section, we hereby focus on the way to further improve the state-of-the-art empirical robust accuracy (ERA) and leave the discussion on improving verifiable robust accuracy (VRA) in Section 5. These two topics are separated because training methods that only enforce empirical robustness are often hard to verify, e.g. even if the point is robustness the certifier may terminate with time-out or an unknown status about the robustness, without calibrating the loss with a particular certification process. This section is organized as follows. In Section 4.1, we discuss several robust training schemes in the prior works. Section 4.2 reveals a bottleneck of these robustness-aware training methods – overfitting to the training data, which is even worse than that in the standard (i.e. non-robust) training. To address the robust overfitting, this thesis proposes a new regularizer for existing robust training losses based on Probably Approximately Correct Bayesian (PAC-Bayesian) bound (Alquier, 2021; Catoni, 2004; Germain et al., 2009; McAllester, 1999; Shawe-Taylor & Williamson, 1997), a learning theory that upper-bounds the loss over distribution with the training loss and an optimizable bound. We provide greater details about our use of PAC-Bayesian bound in Section 4.3 and the resulting method, TrH Regularization, that addresses robust overfitting, in Section 4.4.

4.1 Adversarial Training

The goal of adversarial training is to train a deep network that is robust against ℓ_p norm-bounded noises, i.e. local robustness (Definition 11). Recall that in Chapter 2 we use $f : \mathbb{R}^d \to \mathbb{R}^m$ to denote a neural network that takes an input $x \in \mathbb{R}^d$ and outputs a logit vector for m classes. In addition, we further denote the weights of f as θ so $f(x;\theta_0)$ differ from $f(x;\theta_1)$ by the values of the weights. If we use l to denote a loss function that measures the quality of our prediction, i.e. an identity function, and D^n for a dataset of

n i.i.d instances of (x_i, y_i) sampled from the true data distribution $\mathcal{D}$, the standard training ($\hat{L}$) and test objective (L) are equals to

$$\hat{L}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} l((x,y),\theta),$$

$$\text{and } L(\theta, \mathcal{D}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\, l((x,y),\theta).$$

The standard training and test loss are replaced with their robust counterparts $\hat{R}, R$ in adversarial training as follows

$$\hat{R}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} \max_{||\delta||\leq\epsilon} l((x+\delta,y),\theta),$$

$$\text{and } R(\theta, \mathcal{D}) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \max_{||\delta||\leq\epsilon} l((x+\delta,y),\theta).$$

Because using an identity function for l results in undifferentiable loss, various surrogates can be proposed in the literature. In this thesis, we focus on the following two well-known methods: *adversarial training* (Madry et al., 2017), denote AT (Definition 14), and TRADES (Zhang et al., 2019b) (Definition 15). AT is a classical method in enforcing robustness; while TRADES achieves state-of-the-art robust accuracy on CIFAR-10 and ImageNet (Gowal et al., 2021).

Definition 14 (Adversarial Training (AT) (Madry et al., 2017)) *Given a network f with parameter θ, a training set D^m, the bound of noise ϵ and norm $||\cdot||_p$, AT minimizes the following objective w.r.t θ.*

$$\hat{R}_{AT}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} \max_{||\delta||_p\leq\epsilon} \text{CE}((x+\delta,y),\theta),$$

where $\text{CE}((x+\delta,y)$ *is the Cross Entropy between* $f(x+\delta;\theta)$ *and (the one-hot vector of)* y.

Definition 15 (TRADES (Zhang et al., 2019b)) *With the same assumptions in Definition 14 and let* $\text{softmax}(\cdot), \text{kl}(\cdot||\cdot)$ *be the softmax function and KL Divergence, respectively. TRADES minimizes the following objective w.r.t θ.*

$$\hat{R}_T(\theta, D^n) = \frac{1}{m} \sum_{(x,y)\in D^m} \Big[\text{CE}((x,y),\theta) + \lambda_t \cdot \max_{||\delta||_p\leq\epsilon} \text{KL}((x+\epsilon,x),\theta)\Big],$$

where $\text{KL}((x,x+\delta),\theta) = \text{kl}(s(f(x;\theta))||s(f(x+\delta;\theta))$ *and λ_t is a penalty hyper-parameter, which balances the clean accuracy and the robustness.*

AT is probably the most straight-forward way of encouraging the model to be robust: it adds adversarial noise to the input on the fly and rewards the network if it predicts a correct label for the adversarial example $x+\delta$. Unlike AT, TRADES rewards the network not only for making correct prediction (i.e. the CE part) on the clean input but also for making consistent prediction for all points within a ϵ-ball around

the input (i.e. the KL part). To empirically solve the inner-maximization in both AT and TRADES, the most frequent choices are PGD (Madry et al., 2017) and Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015b; Wong et al., 2020), which take one (i.e. FGSM) or a few steps (i.e. PGD) to maximize the corersponding loss following the direction of gradients and project the perturbation back to the ϵ-ball.

4.2 Overfitting in Adversarial Training

Rice et al. (2020) observe that the AT and TRADES methods may suffer from severe overfitting. Namely, the model exhibits higher robustness on the training data compared to the test data. Overfitting on robustness is even worse than that on the standard accuracy, meaning Rice et al. (Rice et al., 2020) find that the generalization gap of robustness is larger than the that of clean accuracy.

Prior Works for Alleviating Robust Overfitting. A number of traditional strategies have been applied to alleviate the overfitting problem in robust training, such as ℓ_2 weight regularization, early stop (Rice et al., 2020), label smoothing, data augmentation (Yun et al., 2019; Zhang et al., 2017) and using synthetic data (Gowal et al., 2021). Below we describe popular and recent methods that were designed specifically for adversarial training.

- Stochastic Weight Averaging (SWA). Izmailov et al. (Izmailov et al., 2018) introduces SWA for improved generalization performance in standard training. SWA maintains an exponential running average $\theta_{\texttt{avg}}$ of the model parameters θ at each training iteration and uses the running average $\theta_{\texttt{avg}}$ for inference. Namely,

$$\theta^{(t+1)} \leftarrow \texttt{SGD}(\theta^{(t)}); \quad \theta_{\texttt{avg}} \leftarrow \alpha\theta_{\texttt{avg}} + (1-\alpha)\theta^{(t+1)}$$

  where α is commonly set to 0.995. Recent work has also shown that SWA helps generalization in the robust training (Gowal et al., 2021) scenario. Note that maintaining the running average requires additional memory resources.

- Adversarial Weight Perturbation (AWP). Similar to Sharpness-Aware Minimization (Foret et al., 2021), Wu et al. (Wu et al., 2020) encourages model robustness by penalizing against the most offending local weight perturbation to the model. Specifically, given a robust loss $\hat{R}_*(\theta, D^n)$, AWP minimizes the following objective w.r.t θ:

$$\max_{\psi(\xi)\leq\delta_{awp}} \hat{R}_*(\theta+\xi, D^n) \quad (* \text{ denotes either } \texttt{AT} \text{ or } \texttt{T}) \tag{4.1}$$

where ψ measures the amount of noise ξ (e.g. ℓ_2 norm) and δ_{awp} denotes the noise total budget. Note that solving the inner maximization problem requires at least one additional gradient ascent step in the parameter space.

- Second-Order Statistic (S2O). Jin et al. (Jin et al., 2022) improve the robustness generalization by regularizing the statistical correlations between the weights of the network. Furthermore, they provide a second-order and data-dependent approximation A_θ of the weight correlation matrix. During training, S2O minimizes the following objective w.r.t θ:

$$\hat{R}_*(\theta, D^n) + \alpha||A_\theta||_F \quad (* \text{ denotes either } \texttt{AT} \text{ or } \texttt{T}) \tag{4.2}$$

where $\alpha > 0$ is a hyper-parameter and $||\cdot||_F$ is Frobenius Norm. Computing A_θ involves computing the per-instance self-Kronecker product of the logit outputs for the clean and adversarial inputs.

This thesis proposes to alleviate robust overfitting from the perspective of learning theory. That is, the phenomenon of overfitting can be characterized by a PAC-Bayesian bound (Alquier, 2021; Catoni, 2004; Germain et al., 2009; McAllester, 1999; Shawe-Taylor & Williamson, 1997) which upper-bounds the expected performance of a random classifier over the underlying data distribution by its performance on a finite set of training points plus some additional terms. Although several prior works (Gowal et al., 2021; Izmailov et al., 2018; Jin et al., 2022; Wu et al., 2020) have built upon insights from the PAC-Bayesian bound, none attempted to directly minimize the upper bound, likely due to the fact that the minimization of their forms of the PAC-Bayesian bound do not have an analytical solution.

Unlike the PAC-Bayesian bound by Wu et al. (Wu et al., 2020), we rely on a different form of the PAC-Bayesian bound (Germain et al., 2009), which can be readily optimized using a Gibbs distribution (Germain et al., 2016a). In addition, we derive a second-order upper bound over the robust test loss. Interestingly, the resulting bound consists of a regularization term that involves *Trace of Hessian* (TrH) (Ding et al., 2022) of the network weights, a well-known measure of the loss-surface flatness. In the following Section 4.3, we provide greater details on our use of PAC-Bayesian bound to derive a regularization term that better helps to close the robust generalization gap.

4.3 PAC-Bayesian Bound for Robust Generalization

PAC-Bayesian theory (Alquier, 2021; Catoni, 2004; Germain et al., 2009; McAllester, 1999; Neyshabur et al., 2017; Shawe-Taylor & Williamson, 1997) provides a foundation for deriving an upper bound of the generalization gap between the training and test error. Both AWP and S2O plug the robust loss into the classical bound (Neyshabur et al., 2017) to obtain the following form:

Theorem 2 (Square-Root-Form PAC-Bayesian Bound (Wu et al., 2020)) *Given a prior distribution $\mathcal{P}$ on the weight θ of a network, any $\tau \in (0, 1]$, a robustness loss $R(\theta, \mathcal{D})$ for a data distribution $\mathcal{D}$ and its empirical version $\hat{R}(\theta, D^n)$, for any posterior distribution $\mathcal{Q}$ of θ, the following inequality holds with a probability at least $1 - \tau$,*

$$\mathbb{E}_{\theta \in \mathcal{Q}} R(\theta, \mathcal{D}) \leq \mathbb{E}_{\theta \in \mathcal{Q}} \hat{R}(\theta, D^n) + 4\sqrt{\frac{1}{m}\mathrm{kl}(\mathcal{Q}||\mathcal{P}) + \log\frac{2m}{\tau}}. \tag{4.3}$$

Generally speaking, the goal of PAC-Bayesian learning is to optimize $\mathcal{Q}$ on the RHS of Equation 4.3 so as to obtain a tight upper bound on the test error (LHS). However, directly optimizing Equation 4.3 over $\mathcal{Q}$ is difficult because of the square root term. Instead of optimizing the RHS, AWP replaces it with $\max_{\xi} \hat{R}(\theta + \xi, D^n)$ plus a constant upper bound on the square root term, which is only loosely connected to the bound. On the other hand, S2O assumes $\mathcal{P}$ and $\mathcal{Q}$ as non-spherical Gaussian, i.e.

$$\mathcal{P} = \mathcal{N}(0, \Sigma_p), \mathcal{Q} = \mathcal{N}(\theta, \Sigma_q), \tag{4.4}$$

and show that the square root bound increases as the Frobenius norm and singular value of Σ_q increase. To overcome the complexity of the square root term, they approximate it with $\|A_\theta\|_F$ as in Equation 4.2.

As mentioned earlier, neither AWP (Wu et al., 2020) nor S2O (Jin et al., 2022) minimizes the PAC-Bayesian bound directly, likely due to the complicating square-root term in Equation 4.3. However, Equation 4.3 is only one instance out of PAC-Bayesian bounds defined in Germain et al. (Germain et al., 2009). According to their framework, there is also a *linear-form* of the bound related to Bayesian inference which can be analytically minimized with the aid of a Gibbs distribution (Ding et al., 2022; Germain et al., 2016b; Rothfuss et al., 2021). We leverage this linear-form of the PAC-Bayesian bound, and adapt it for bounding the robustness gap:

Theorem 3 (Linear-Form PAC-Bayesian Bound (Germain et al., 2016b)) *Under the same assumptions as in Theorem 2, for any $\beta > 0$, with a probability at least $1 - \tau$,*

$$\mathbb{E}_{\theta \in \mathcal{Q}} R(\theta, \mathcal{D}) \leq \ \mathbb{E}_{\theta \in \mathcal{Q}} \hat{R}(\theta, D^n) + \frac{1}{\beta}\mathrm{kl}(\mathcal{Q}||\mathcal{P}) + C(\tau, \beta, m), \tag{4.5}$$

where $C(\tau, \beta, m)$ is a function independent of $\mathcal{Q}$.

Here β is a hyper-parameter that balances the three terms on the RHS. A common choice is $\beta \propto m$, i.e. the size of the dataset (Ding et al., 2021; Germain et al., 2016b). Recently, (Ding et al., 2022) proposed to set $\mathcal{P}$ and $\mathcal{Q}$ to univariate Gaussians and used a second-order approximation to estimate the PAC-Bayesian bound. Applying a similar technique, we introduce Theorem 4 (see Appendix. D.1 for the proof) which minimizes the RHS of Inequality 4.5.

Theorem 4 (Minimization of PAC-Bayesian Bound) *If $\mathcal{P} = \mathcal{N}(\mathbf{0}, \sigma_0^2)$, and $\mathcal{Q}$ is also a product of univariate Gaussian distributions, then the minimum of Inequality 4.5 w.r.t $\mathcal{Q}$ can be bounded by*

$$\begin{aligned}\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D}) &\leq \min_{\mathcal{Q}} \{\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^n) + \frac{1}{\beta}\mathrm{kl}(\mathcal{Q}||\mathcal{P})\} + C(\tau, \beta, m)\} \\ &= \min_{\theta} \{\hat{R}(\theta, D^n) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2}\mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^n))\} + C(\tau, \beta, m) + O(\sigma_0^4). \end{aligned} \tag{4.6}$$

Assuming that $\sigma_0^2 = 1/d$ is the variance of the initial weight matrices where d is the input feature dimension, then $O(\sigma_0^4) = O(d^{-2})$. Theorem 4 considerably simplifies the optimization problem from one over the space of probability density functions $\mathcal{Q}$ to one over model weights θ. Moreover, the resulting bound (RHS of Eq. 4.6) contains the Trace of Hessian (TrH) term $\mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^n))$ of the robust loss, which sums the loss surface curvatures in all directions. For a convex loss, the Hessian is positive semi-definite (PSD) in which case trace minimization leads θ to a flatter region of the loss surface. Although in general, our Hessian is not PSD, empirically we find that its largest eigenvalue has a much larger magnitude than the least ones (Alain et al., 2018) and that trace minimization correlates with minimizing the standard deviation of the eigenvalues so that the overall curvature is effectively decreases (Example 1 to follow in Section 4.4).

Remark 1 (Relation Between $\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D})$ and $\min_\theta R(\theta, \mathcal{D})$) *Compared to $\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D})$ where we minimize the mean and variance of the Gaussian posterior distribution family, directly minimizing the test loss over a* deterministic *θ is probably more related to one's goal of training. Since the PAC-Bayesian bound only captures the behavior of a distribution, it is technically unreliable with the current bound, i.e. Theorem 4, used in this thesis. However, we notice that the optimal variance is given in the proof of Theorem 3 (Appendix **??**), which takes the form of,*

$$\sigma_n^{2*} = \left(\frac{1}{\sigma_0^2} + \beta\frac{\partial^2 \hat{R}}{\partial \theta_n^2}\right)^{-1}.$$

Since β is proportional to the number of data examples m, the variance σ_n^{2*} becomes very small for any reasonably sized datasets (e.g. $m = 60K$ in CIFAR, $m = 1M$ in ImageNet). The resulting posterior, in practice, is a "narrow" Gaussian. So one can simply take its optimal mean as a point estimator of the true posterior, i.e. $\min_\theta R(\theta, \mathcal{D})$, during the test time.

The RHS of Eq. 4.6 in Theorem 4 provides a training objective function that bounds the optimal test error up to a constant. However, evaluating and minimizing the bound in Eq. 4.6 requires computing the Trace of Hessian (TrH) term. Unfortunately, computing the Hessian directly by applying auto-differentiation twice is infeasible for a large deep network. Another solution is to apply Hutchinson's method (Avron &

Toledo, 2011) to randomly approximate the TrH, however, the variance of the resulting estimator is too high and it ends up being too noisy for training in practice. Instead, we propose to restrict TrH regularization to the top layer of the deep network only. Although this restriction brings approximation error to the bound, it has two benefits. First, the TrH on the top layer has simple analytical expressions (details to follow in Section **??**). Second, we show that the top-layer TrH regularization effectively regularizes the TrH of the entire network as well (details to follow in Section **??**).

4.4 Top-layer TrH Regularization for AT and TRADES

Recall that for a network f we use θ to denote its parameter. In this section, we additionally use the following decomposition, i.e.

$$\theta = \{\theta_t, \theta_b\},$$

to separately denote θ_t as the weights of the top layer and θ_b as the weights of the remaining (lower) network. In this case, the logits of f can thus be expressed as

$$f(x;\theta) = f(f(x;\theta_b);\theta_t) = \theta_t^\top f(x;\theta_b)$$

where $f(x;\theta_b)$ is the penultimate layer feature. Furthermore, we define

$$h(x;\theta) = \text{softmax}(f(x;\theta)) - \text{softmax}(f(x;\theta))^2.$$

Now, we present the analytical forms of the top-layer TrH w.r.t the AT and the TRADES in Proposition 2 and 3.

Proposition 2 (TrH for AT) *Given a training dataset D^n and the adversarial input example x' for each example x, the top-layer* TrH *of the AT loss (Definition 14) is equal to*

$$\text{Tr}(\nabla^2_{\theta_t} \hat{R}_{AT}(\theta, D^n)) = \frac{1}{m} \sum_{(x,y)\in D^m} \text{TrH}_{AT}(x';\theta),$$
$$\textit{where} \quad \text{TrH}_{AT}(x';\theta) = ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x';\theta).$$

Proposition 3 (TrH for TRADES) *Under the same assumption in Proposition 2, the top-layer TrH of the TRADES loss (Definition 15) is equal to*

$$\text{Tr}(\nabla^2_{\theta_t} \hat{R}_T(\theta, D^n)) = \frac{1}{m} \sum_{(x,y)\in D^m} \text{TrH}_T(x, x';\lambda_t,\theta),$$
$$\textit{where,} \quad \text{TrH}_T(x, x';\lambda_t,\theta) = ||f(x;\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x;\theta) + \lambda_t ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x';\theta). \tag{4.7}$$

Algorithm 1: TrH Regularization for Training Adversarial Robust Network

Hyper-parameters: The ℓ_2-norm penalty for weights γ, the TRADES penalty λ_t and the TrH penalty λ.
Inputs: A batch B of examples, a `loss_type` ='`AT`' or '`TRADES`', the noise budget δ, and a network $f(x;\theta) = \theta_t^\top f(x;\theta_b)$.
Output: The training objective at the current iteration.

$R \leftarrow 0$;
foreach $(x, y) \in B$ **do**
  $x' \leftarrow$ `PGD_InnerLoop`$(x, y, \delta,$ `LossType`$)$
  $z' \leftarrow f(x';\theta_b)$, $h' \leftarrow \text{softmax}(\theta_t^\top z') - \text{softmax}(\theta_t^\top z')^2$; `// This step computes the penultimate feature` z' `and the softmax gradient` h' `on the adversarial input` x'.
  if *LossType* == '*AT*' **then**
    $R \leftarrow R + \text{CE}((x', y), \theta) + \lambda(||z'||_2^2 \cdot \mathbf{1}^\top h')$;
  else if *LossType* == '*TRADES*' **then**
    $z \leftarrow f(x;\theta_b)$, $h \leftarrow \text{softmax}(\theta_t^\top z) - \text{softmax}(\theta_t^\top z)^2$; `// TRADES needs the` z `and` h `from the benign` x.
    $R \leftarrow R + \text{CE}((x, y), \theta) + \lambda_t \text{kl}(\text{softmax}(\theta_t^\top z)||\text{softmax}(\theta_t^\top z')) + \lambda(||z||_2^2 \cdot \mathbf{1}^\top h + \lambda_t||z'||_2^2 \cdot \mathbf{1}^\top h')$;
  end
end
return $\frac{1}{|B|} R + \gamma||\theta||_2^2$

To obtain Eq. 4.7, we stopped the gradient on the KL in Definition 15 with respect to the clean logits $f(x;\theta)$, because we find it is more stable for training, otherwise, there would be an additional regularization term $G(x, x';\theta)$ in the TrH_T (see more details in the Appendix D.1). With Proposition 2 and 3, we now present the complete training objective in Algorithm 1. Notice that, to simplify hyper-parameter notations, we re-parameterize $\gamma \stackrel{\text{def}}{=} 1/2\beta\sigma_0^2$ and $\lambda \stackrel{\text{def}}{=} \sigma_0^2/2$ in the algorithm.

Remark 2 (Compared to Sharpness-Aware Regularization (SAM) (Foret et al., 2021)) *Although related, our work differs from SAM (or AWP) with regards to how PAC-Bayesian theory and sharpness are positioned. SAM is motivated by empirically reducing the sharpness of the training loss landscape, whereas our work is theoretically motivated and directly minimizes the PAC-Baysian bound of the (adversarial) loss w.r.t the Gaussian posterior so as to reduce the generalization error. Specifically, the sharpness measure* TrH *emerges as a result of that optimization. Although the PAC-Bayesian bound was also considered as a theoretical motivation for SAM, it is unclear whether SAM optimization indeed reduces the PAC bound. For example, Wen et al. (Wen et al., 2022) shows that SAM is equivalent to minimize the trace of Hessian (TrH) when batch size is 1, whereas our Theorem 4 arrives at* TrH *minimization regardless of the batch size.*

Although our TrH regularization is restricted to the top layer only, the gradient back-propagates to the entire network through the penultimate layer features. This motivates the following research question:

What effect does top-layer TrH regularization have on the TrH of the entire network?

To answer this question, we take a two-step approach. First, we experiment with a small 3-layer network on the synthetic Two-Moon dataset (Figure 4.1), provided that computing the full network Hessian is

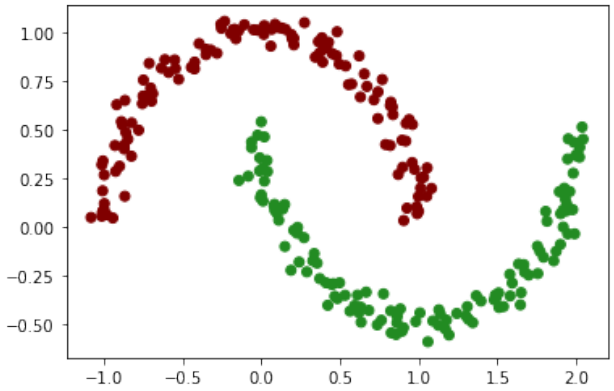


Figure 4.1: Visualization of Two Moons dataset. The task is to classify the points into two classes (red and green).

inexpensive. Subsequently, we provide a theorem that can be used to bound the Trace of Hessian of the full network with that of the top layer only.

Example 1 (Two Moons) *We use* 500 *training examples, where* $x \in \mathbb{R}^2$ *and* $y \in \{0, 1\}$ *from the Two Moon dataset (Figure 4.1). We train a small network with the AT loss using the following settings:*

- ***Network.*** *We use a dense network Dense(100)-ReLU-Dense(100)-ReLU-Dense(2).*
- ***Training without Regularization (Standard).*** *We train the network with AT(base), such that it is robust in an* ℓ_∞ *ball of size 0.02 with 1 PGD step. We use a momentum-SGD with learning rate 0.1 for 100 epochs.*
- ***Training with Top-Layer*** TrH ***Regularization (Top).*** *We further add the* TrH *of the top layer as a regularization term in the loss. We compute the* TrH *at the top layer using Proposition 2. The regularizing coefficient for the top-layer TrH is 0.5 (i.e.* $loss = AT + 0.5 * \text{TrH}_{top}$*) as we find it needs a stronger penalty due to the lack of* TrH *from other layers.*
- ***Training with Full*** TrH ***Regularization (Full).*** *In additional to the Standard setup, we add the TrH of the full network as a regularization term in the loss. We numerically compute the Hessian and its trace. The regularizing coefficient for the full-network* TrH *is 0.05 (i.e.* $loss = AT + 0.05 * \text{TrH}_{full}$*).*

Throughout the training process (x-axis), we evaluate the TrH *of the* entire *network (y-axis) with a numerical approach (twice differentiation) and plot the sum of Hessian eigenvalues, i.e.* TrH*, and the standard deviations of eigenvalues in Figure 4.2.*

Focusing on the top-left plot in Figure 4.2, we see that the TrH decreases across time with standard training (blue), but at the same time, the full-network and top-layer TrH regularization decrease it even further. In particular, top-layer TrH regularization (orange) is nearly as effective as directly regularizing

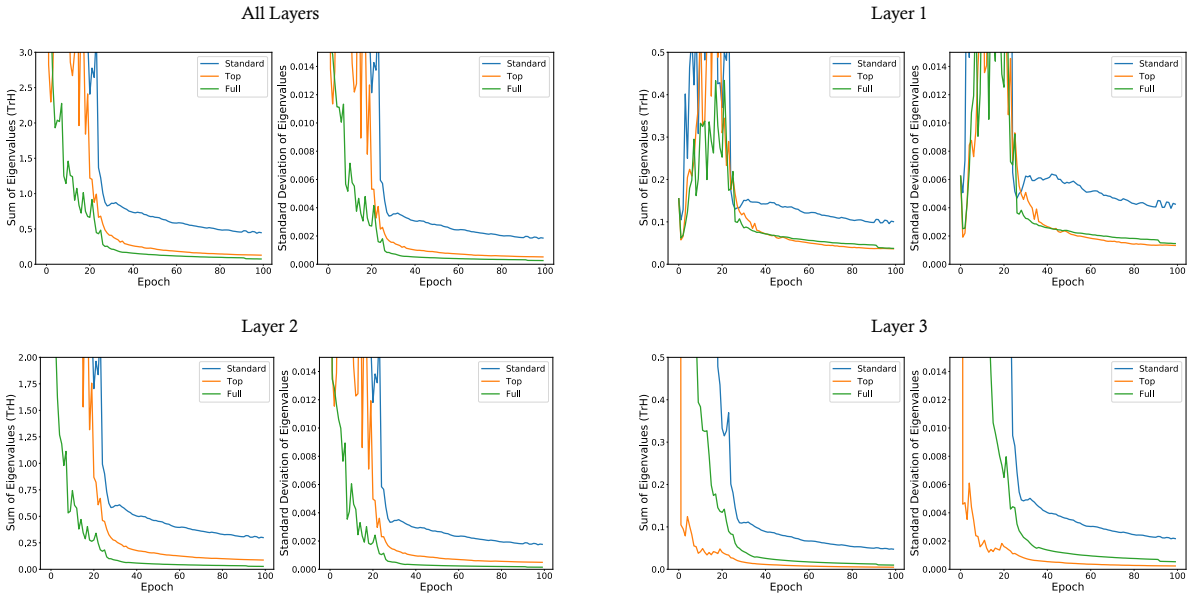


Figure 4.2: The figure shows pairs of plots for the sum and the standard deviation of the Hessian matrix eigenvalues. The top-left pair corresponds to the Hessian of all layers, while the rest to each of the three layers separately, with Layer 3 being the top layer (note that the sum of eigenvalues is exactly the trace of Hessian). As can be seen on the top-left , the sum and standard deviation decrease in standard training (blue) and moreover, this effect is amplified by direct TrH regularization with similar results for full network regularization (green) and top-layer regularization (orange). This similarity can be explained by our Theorem 5

the TrH for all layers (green) after the 60-th epoch. The standard deviation plots (on the right side of each TrH plot) show a decrease in the standard deviation, which implies there is a contraction effect on the eigenvalues of the Hessian, with both positive and negative values approaching towards 0. This rules out the possibility that the TrH reduction comes from an increase in the magnitude of the negative eigenvalues. In fact, TrH regularization effectively contracts the magnitude of all eigenvalues and leads to a smoother region in the loss surface. We further plot the TrH and standard deviation of Hessian eigenvalues for each individual layer respectively (where Layer 3 is the top layer). Except for the first layer, whose eigenvalues seem to be small from the onset, the eigenvalue contraction effect is evident as training progresses, with a stronger and similar effect for both TrH regularization settings.

We provide Example 1 as an empirical justification of our use of top-layer TrH regularization. Next, we provide a theoretical justification for the impact of top-layer TrH regularization on the layers below. Intuitively, our Theorem 5, establishes an inductive relation between the TrH of the consecutive layers in a feedforward neural network with ReLU activation and CE loss, a common building block in AT and TRADES training.

Theorem 5 (Inductive Relation in TrH) *Suppose $f : \mathbb{R}^d \to \mathbb{R}^m$ is a feed-forward network with ReLU activation. For the i-th layer, let $W^{(i)}$ be its weight and $\mathcal{I}_x^{(i)}$ be its input evaluated at x. Thus, $\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)})$, i.e. TrH evaluated at x using a CE loss w.r.t $W^{(i-1)}$, is equal to*

$$\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)}) = ||\{\mathcal{I}_x^{(i-1)}\}||_2^2 \sum_{k,d \in P^{(i)}} \{\mathcal{H}^{(i)}\}_{k,d}$$

$$\textit{where } \{\mathcal{H}^{(i)}\}_{k,d_i} = \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]^2 \cdot \{h(x,\theta)\}_k$$

$$P^{(i)} = \{d_i | \{\mathcal{I}_x^{(i)}\}_{d_i} > 0\}$$

and the following inequality holds for any $\mathcal{H}^{(i)}, \mathcal{H}^{(i+1)}$:

$$\max_{k,d_i} \{\mathcal{H}^{(i)}\}_{k,d_i} \leq \max_{k,d_{i+1}} \{\mathcal{H}^{(i+1)}\}_{k,d_{i+1}} \cdot ||W^{(i)}||_1^2. \tag{4.8}$$

Put simply, the max-norm of the Hessian of each layer forms an upper bound for the max-norm of the layer below it (times a scalar). Therefore, regularizing the TrH of the top layer alone implicitly regularizes the Hessian of the layers below it. This effect explains the phenomenon observed in Example 1.

Furthermore, the operator norm $||W^{(i)}||_1^2$ of weights in Eq. 4.8 makes an interesting connection between TrH minimization and robust training. Roth et al. (Roth et al., 2020) show that robust training is an implicit way of regularizing the weight operator norm. Moreover, directly minimizing the operator norm is a commonly used technique in training certifiable robust networks (Huang et al., 2021a; Leino et al., 2021b), a stronger version of the robustness guarantee targeted in this paper. Thus, the combination of robustness training and top-layer regularization results in an implicit regularization on TrH for internal layers.

4.5 Evaluations

In this section, we evaluate TrH regularization (Algorithm 1) over three image classification benchmarks including CIFAR-10/100 and ImageNet and compare against the robust training baselines, i.e. SWA (Gowal et al., 2021; Izmailov et al., 2018), AWP (Wu et al., 2020), and S2O (Jin et al., 2022), described in Section 4.2. We begin with the setups for our experiments.

4.5.1 Experiment Setup

Datasets and Threat Model. For CIFAR-10/100, we choose the standard $\epsilon = 0.031 (\approx 8/255)$ of the ℓ_∞ ball to bound the adversarial noise. In addition, Gowal et al. (Gowal et al., 2021) has reported that using synthetic images from a DDPM (Ho et al., 2020) generator can improve the robustness accuracy for

the original test set. Thus, we also add 1M DDPM images to CIFAR-10/100 (these images are publicly availabe (dee)). For ImageNet, we choose $\epsilon = 3.0$ for the ℓ_2 and $\epsilon = 0.0157(\approx 4/255)$ for the ℓ_∞ ball, respectively. We do not use extra data for ImageNet. The chosen ϵs follow the common choices in the literature (Gowal et al., 2021; Jin et al., 2022; Madry et al., 2017; Wu et al., 2020). We report the test accuracy evaluated with benign images and adversarial images generated by AutoAttack (Croce & Hein, 2020a) (AutoPGD+AutoDLR), which is widely used in the literature (Gowal et al., 2021; Jin et al., 2022; Wu et al., 2020).

Network Architectures. We use Vision Transformer (ViT) (Dosovitskiy et al., 2021) through all experiments because of its success on multiple tasks over existing architectures. Specifically, we report the results of ViT-L16, Hybrid-L16 for CIFAR-10/100 and ViT-B16, ViT-L16 for ImageNet (additional results and architecture details can be found in Appendix D.5). We initialize the weights from a pre-trained checkpoint on ImageNet21K (goo). Notice that the resolution of the CIFAR images (32×32) are too small to correctly align with the pre-trained kernel of the first layer. To address this issue, we down-sample the kernel of the input layer following the receipt proposed by Mahmood et al. (Mahmood et al., 2021).

Methods. We train the models using AT and TRADES losses, together with the following defenses: (1) base: no regularization; (2) AWP (Wu et al., 2020); (3) SWA (Izmailov et al., 2018); (4) S2O (Jin et al., 2022); and (5) our Algorithm 1 (TrH). We are interested in the baselines defenses because of their connections to the PAC-Bayesian bound or the loss flatness. During training, the model was partitioned over four Google Cloud TPUv4 chips (Jouppi et al., 2017) for the base and TrH methods, and eight chips for AWP, S2O, and SWA as they consume more HBM memory.

Hyperparameter Selection. Training robust ViTs can be challenging due to a large number of hyper-parameters. We use two steps for hyperparameter selection. For the choice of common hyper-parameters, e.g. batch size, patch size, training iterations, and etc. (a full list is included in Appendix D.2), we first do a large grid search and select values that produce the best results on TRADES(base). This step is referred to as pre-tuning and is done per dataset. After the first step, where the common hyper-parameters are locked, we tune and report the best results after trying different sets of method-specific hyper-parameters, e.g., the norm bound of weight noises in AWP and λ for TrH. Appendix D.3 provides a full list of hyper-parameters we use to produce our main results in Table 4.1.

$\ell_\infty(\epsilon = 8/255)$	ViT-L16				Hybrid-L16			
SE= ±0.5%	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(base)	87.4	60.8	64.3	31.7	88.0	**61.7**	64.2	31.8
AT(SWA)	88.0	61.7	62.5	31.7	86.5	57.2	63.6	30.8
AT(S2O)	87.1	60.2	64.9	31.7	88.7	61.6	64.3	31.8
AT(AWP)	88.5	**62.4**	63.3	32.6	88.5	61.5	63.9	**32.5**
AT(TrH)	87.0	61.7	62.5	**32.8**	88.0	61.0	63.8	32.4
TRADES(base)	85.2	60.3	62.0	31.4	85.9	60.8	61.3	30.6
TRADES(SWA)	85.9	60.7	61.5	32.0	84.3	59.4	62.3	29.9
TRADES(S2O)	87.4	**61.6**	62.4	31.6	87.2	61.5	65.0	31.9
TRADES(AWP)	85.3	60.8	63.0	**32.2**	84.5	59.8	61.4	32.1
TRADES(TrH)	86.4	61.4	62.0	32.1	87.4	**61.7**	66.4	**34.1**

ImageNet	ViT-B16				ViT-L16			
SE= ±0.2%	$\ell_\infty(\epsilon = 4/255)$		$\ell_2(\epsilon = 3.0)$		$\ell_\infty(\epsilon = 4/255)$		$\ell_2(\epsilon = 3.0)$	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(base)	72.2	39.0	71.0	39.3	75.4	44.1	73.9	43.5
AT(SWA)	72.3	40.0	71.5	39.8	75.2	44.3	74.3	43.8
AT(S2O)	72.0	39.0	71.4	39.3	75.4	46.2	76.3	46.4
AT(AWP)	71.3	39.5	70.5	39.3	74.5	44.0	73.8	44.0
AT(TrH)	71.3	**42.2**	71.6	**42.4**	75.8	**48.8**	74.2	**47.0**
TRADES(base)	69.2	39.3	67.8	39.7	77.8	40.9	77.1	41.5
TRADES(SWA)	69.5	39.7	68.1	40.0	72.7	44.5	71.1	44.2
TRADES(S2O)	70.6	39.3	69.8	39.7	73.7	43.6	72.5	44.0
TRADES(AWP)	65.8	38.6	64.5	38.3	68.5	43.2	67.2	42.7
TRADES(TrH)	65.3	**41.9**	66.4	**41.6**	70.1	**48.9**	68.9	**47.3**

Table 4.1: Clean: % of Top-1 correct predictions. ERA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) Stark & University of California (2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (n as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

4.5.2 Main Results.

In Table 4.1, we report the robust test accuracy using different types of defenses. The top results are highlighted in bold font. To measure the significance of an improvement, we calculate the maximum standard error (SE) (Stark & University of California, 2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (where n denotes the number of test examples) for each dataset. Thus, the accuracy of a certain model is regarded a *silver* result a_{silver}, if its SE interval overlaps with the that of the top result a_{top}. Namely, $a_{top} - SE \leq a_{silver} + SE$.

On CIFAR-10/100, a significant improvement is at least 1% according to SE. Therefore, based on Table 4.1, the performance differences among the methods are relatively small. Nevertheless, TrH attains either the top result (in 3 instances) or the silver result (in 5 instances) across all eight instances. In comparison, AWP is the second best with 3 top and 3 silver; S2O has 1 top and 4 silver; SWA has 3 silver; and the base has 1 top and 3 silver. Notably, Hybird-L16+TRADES(TrH) achieves the highest robust

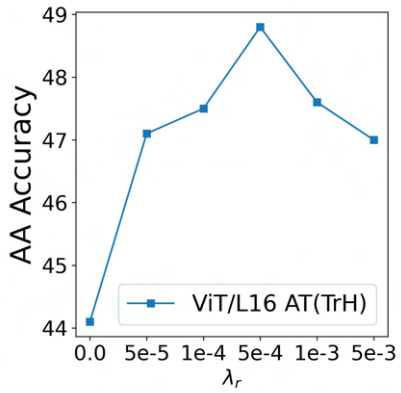


Figure 4.3: A plot of the Autoattack accuracy (ERA %) against λ (TrH Penalty coefficient) when training on ImageNet with TrH Regularization in the ℓ_∞ case.

accuracy (34.1%) on CIFAR-100 which beats all other baselines by at least 2%.

On ImageNet, a significant improvement is at least 0.4% according to SE. We observe that TrH regularization outperforms the other baselines across the board. Using AT(TrH), we set a new *state-of-the-art* robust accuracy of 48.8%, at $\ell_\infty(4/255)$ and 47.0% at $\ell_2(3.0)$ with a ViT-L16 model. Specifically, in the case of ℓ_∞, 48.8% is an improvement of 4.7% compared to the basic setup AT(base). In TRADES(TrH), although the robust accuracy is even slightly higher than AT, the clean accuracy is lower. This is due to the choice of $\lambda_t = 6$, which strongly favors robustness over clean accuracy. As we reduce λ_t to 1, we obtain a better balance of (clean, ERA) accuracy of TRADES(TrH) at (78.7, 46.7) for l_∞ and (77.1, 45.2) for l_2.

Summary of Results in Table 4.1 The results in Table 4.1 demonstrate that training with TrH regularization either matches or outperforms the robust accuracy of the existing methods. In fact, the gains of TrH regularization are wider on the larger ImageNet dataset compared to on CIFAR-10/100, where several methods show equivalent performance (per Standard Error analysis). This suggests that future work in robust adversarial training should move beyond the CIFAR benchmarks to larger-scale tasks.

Sensitivity to λ. To study the sensitivity of λ to the training in TrH regularization, we train a ViT-L16 model over the ImageNet dataset and vary the choice of λ. Figure 4.3. plots the ERA accuracy (y-axis) against the different choices of λ (x-axis) with all other setups fixed. The plot shows that $\lambda = 5 \times 10^{-4}$ attains the highest AutoAttack accuracy (i.e. ERA) for both AT and TRADES losses with little degradation in the range of $[10^{-4}, 10^{-3}]$.

Training Efficiency. To compare the computational efficiency of the various methods, we report the peak memory usage and runtime performance (i.e., images processed per second; the higher the better) in

Defense	Mem.(MB/chip)	Img/sec/chip
AT(base)	5.0×10^3	5.5
AT(TrH) (ours)	5.0×10^3	5.2
AT(SWA)	5.6×10^3	4.4
AT(S2O)	8.0×10^3	5.2
AT(AWP)	6.8×10^3	3.7

Table 4.2: Peak memory usage and run-time reports measured when training CIFAR-10 using a ViT-L16 with a batch size of 32. Training is paralleled on two NVIDIA RTX chips. Lower memory usage and higher img/sec/chip are more efficient.

$\ell_\infty(\epsilon = 8/255)$	ViT-L16				Hybrid-L16			
SE= ±0.5%	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(AWP)	88.5	**62.4**	63.3	32.6	88.5	61.5	63.9	32.5
AT(AWPT)	88.9	62.2	63.9	31.9	87.5	61.3	63.3	**33.0**
AT(TrH)	87.0	61.7	62.5	**32.8**	88.0	61.0	63.8	32.4
TRADES(AWP)	85.3	60.8	63.0	**32.2**	84.5	59.8	61.4	32.1
TRADES(AWPT)	85.3	60.5	62.0	**32.2**	85.8	61.4	65.7	33.8
TRADES(TrH)	86.4	61.4	62.0	32.1	87.4	**61.7**	66.4	**34.1**

ImageNet	ViT-B16				ViT-L16			
SE= ±0.2%	$\ell_\infty(\epsilon = 4/255)$		$\ell_2(\epsilon = 3.0)$		$\ell_\infty(\epsilon = 4/255)$		$\ell_2(\epsilon = 3.0)$	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(AWP)	71.3	39.5	70.5	39.3	74.5	44.0	73.8	44.0
AT(AWPT)	71.6	40.6	71.0	40.0	75.2	44.5	74.1	44.2
AT(TrH)	71.3	**42.2**	71.6	**42.4**	75.8	**48.8**	74.2	**46.7**
TRADES(AWP)	65.8	38.6	64.5	38.3	68.5	43.2	67.2	42.7
TRADES(AWPT)	68.0	39.8	67.0	39.7	70.5	44.9	68.9	44.9
TRADES(TrH)	65.3	**41.9**	66.4	**41.6**	70.1	**48.9**	68.9	**47.3**

Table 4.3: Comparisons between AWPT, AWP and TrH regularizations. Clean: % of Top-1 correct predictions. ERA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) Stark & University of California (2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (i.e. n as the number of test images) is computed for each dataset.

Table 4.2. As the cloud TPU chips are dynamically allocated and preemptable, we instead measure these costs on two local NIVIDA RTX chips with 24G memory per chip. Because by default JAX preallocates all GPUs, we set `XLA_PYTHON_CLIENT_ALLOCATOR=platform` before measuring the memory allocation and run-time. In Table 4.2, we show that adding TrH regularization brings almost no additional memory usage and runtime slowdown, compared to training without any regularization.

4.5.3 On the Promise of Top-Layer Regularization

Our TrH regularization demonstrates that by only regularizing the weights on the top layer it provides useful gradient information for the bottom network to update the weights to learn better representations.

Similarly, we can also modify AWP to only work on the top layer and we name this appproach as AWPT (i.e. T stands for "on the top"). The fine-tuning setup is identical to our setup in AWP training and we present the results in Table 4.3. Interestingly, we find retaining the AWP step only to the top layer does not hurt the final performance compared to AWP and in some cases, i.e. on CIFAR-100 using a Hybrid-L16 model, AWPT is even better than AWP. This inspires a few research questions such that

do we actually need to regularize the bound for the entire network?

From our proposed work on TrH regularization and AWPT, we see that the answer to that question may be NO. Our findings should inspire future work to discuss even more efficient ways of improving or matching the current robust accuracy.

4.6 Related Work

In this section, we situate our contributions in the broader context of current literature, such as related work on PAC-Bayesian theory, flatness regularization, as well as other types of robust adversarial losses.

PAC-Bayesian Theory and Flatness. Beyond the PAC-Bayesian inspired approaches for robust training that were already mentioned in the paper (such as Wu et al. (Wu et al., 2020)), a different line of work focuses on adapting PAC-Bayesian bounds for an ensemble of models, to improve either adversarial robustness (Viallard et al., 2021) or out-of-distribution (OOD) robustness data (Zecchin et al., 2022). Several other works have focused on methods for finding flat local minima of the loss surface in order to improve (standard) model performance(Ding et al., 2022; Foret et al., 2021; Jia & Su, 2020). Also, outside the scope of adversarial robustness, Ju et al. (Ju et al., 2022) find that trace of Hessian regularization leads to robustness against noise in the *labels*. Their Hessian regularization is based on randomized estimation and is done for all layers, whereas we effectively apply TrH regularization on the top layer only and with Theorem 5 as theoretical justification.

Perhaps the closest approach to ours is the PACTran-Gaussian metric (Ding et al., 2022), which uses a 2nd-order approximation of the PAC-Bayesian bound as the metric for evaluating the transferability of pretrained checkpoints. Similarly to our Eq. 4.6, the PACTran metric is composed of two terms: an ℓ_2 regularized empirical risk (RER) and a flatness regularizer (FR). Our work was inspired by the PACTran metric, but also makes several improvements. Firstly, in PACTran a single posterior variance σ_q^2 was shared for all parameters. On the contrary, we used different σ_q^2's for different parameters. Secondly, in PACTran the optimization of the posterior mean θ_q is over the RER term only, while our optimization is over both RER and FR. These two changes potentially make our bound tighter than the one in the PACTran paper.

In addition, we also study and explicitly include the high-order terms in $O(\sigma_0^4)$, which was neglected in PACTran.

Adversarial Surrogate Loss. In this paper, we focused only on the AT and TRADES approaches for robust adversarial optimization as they achieve reasonable results on the datasets of choice. In fact, recent work (Gowal et al., 2021) on achieving state-of-the-art robustness on CIFAR-10 also exclusively focuses on AT or TRADES. Beyond AT and TRADES, the works in (Ding et al., 2020; Pang et al., 2022; Wang et al., 2020b) explore other types of adversarial surrogate losses. Our method can be easily transferred to these losses and we include additional examples in Appendix D.1.4.

4.7 Chapter Summary and Future Work

In Chapter 4, we discuss a novel way to improve the empirical robustness of deep networks. In particular, we alleviate the overfitting in adversarial training with PAC-Bayesian theory. Using a linear-form PAC-Bayesian bound overlooked by the prior work, we derive a 2nd-order estimation that includes the Trace of Hessian (TrH) of the model. We then focus on the TrH of the top layer only, showing both empirically and theoretically (for ReLU networks) that its minimization effectively leads to the minimization of the TrH of the entire network, thereby providing a sound flatness regularization technique for adversarial training which adds only little computational overhead. Experimental results show that our method is consistently among the top performers on CIFAR-10/100 benchmark as well as achieves a significant improvement in robust test accuracy on ImageNet compared to the other baselines.

The success of TrH regularization on the top layer, on the other hand, should inspire a closer look at the marginal outcome of many full-network regularizations. Namely, does the regularization term provides a necessary better gradient direction for the training by taking all parameters from the network, compared to using only a fraction of the parameters? As modern architecture always grows up likely in an exponential fashion, implementing a full-network regularization can be expansive in practice. For resource-constraint scenarios, we have been faced with a trade-off between benefiting from an insightful theorem and the cost to realize it. Our top-layer regularization inspires this novel direction of only restricting the classification layer on the top and let the rest of the network to learn corresponding representations that will calibrate with the dynamic of the top layer. In practice, this top-layer idea not only works for our TrH regularization but also works for AWP as is shown in Section 4.5.3, pointing out an under-explored relation between the behavior of a fraction of the network and the whole.

Chapter 5

Training Provably Robust Networks

One downside of evaluating models with an empirical attack and reporting the ERA as is done in Chapter 4 is that it only provides an upper-bound of the error that the model makes in the test time. The state-of-the-art attack used in Chapter 4, at the time the thesis is written, will get replaced by even stronger and well-designed attacks in the future so our evaluation may become less meaningful in the future's hindsight. One solution to the problem above is to train a model that can *certify* its predictions within an ϵ-ball, which is, therefore, the focus of this chapter. Namely, a classifier $F : \mathbb{R}^d \to [m]$ with certification can be written as $F^\epsilon : \mathbb{R}^d \to [m] \cup \{\perp\}$ such that

$$F^\epsilon(x) = \begin{cases} F(x) & \text{if } F(x) \text{ is } \epsilon\text{-local robust at } x \text{ (Definition 11)} \\ \perp & \text{otherwise} \end{cases}.$$

Compared to a conventional network F, F^ϵ opts to abstain the prediction by returning $\perp$ if the input is not robust, i.e. less than ϵ-away from the decision boundary (see an illustration in Figure 5.1). Classification with an option to abstain naturally happens to the case when more than one objects appear in the image, e.g. an animal classifier cannot robustly classify an input to one class with both cats and dogs captured in the image.

Over the last few years, a wide body of literature addressing robustness certification has emerged to check for the local robustness of $F(x)$ (Cohen et al., 2019a; Croce et al., 2019; Fromherz et al., 2021b; Huang et al., 2021b; Jordan et al., 2019b; Lee et al., 2020b; Leino et al., 2021d; Li et al., 2019a,b; Singla et al., 2022; Tjeng et al., 2019a; Trockman & Kolter, 2021; Wong & Kolter, 2018a). To date, the methods that achieve by far the best certified performance are derived from *randomized smoothing* (Cohen et al., 2019a); however, this provides only a *probabilistic* guarantee—which may generate a false positive claim around 0.1% of the time (Cohen et al., 2019a)—and requires significant overhead for both evaluation and certification. By contrast, *deterministic* certification may be preferred in safety-critical applications, e.g.,

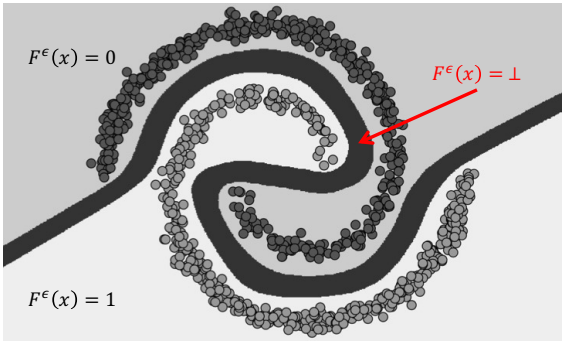


Figure 5.1: A 2D example of a binary classifier with abstention F^ϵ.

malware detection and autonomous driving. Unless noted otherwise, the rest of this chapter should concentrate on deterministic certification.

Because of the highly non-linear boundaries learned by a neural network, deterministically certifying the robustness of its predictions usually requires specialized training procedures that regularize the network for efficient certification, as post-hoc certification is either too expensive (Katz et al., 2017; Sinha et al., 2018) or too imprecise (Fromherz et al., 2021b), particularly as the scale of the model being certified is increased. The most promising such approaches—in terms of both certified accuracy and efficiency—perform certification using Lipschitz bounds, meaning that the learning procedure must impose Lipschitz constraints on the layers during training, either through regularization (Leino et al., 2021d) or orthogonalization (Trockman & Kolter, 2021).

This chapter introduces a novel kind of model architecture that is constructed with an option to abstain with a Lipschitz-based certification process, which we term as Globally Robust (GloRo) Nets. Compared to existing methods, the certification process is *instant* in GloRo Nets, e.g. 600x times faster than dual network approaches (Wong & Kolter, 2018b) and 10^5x faster than randomized smoothing (Cohen et al., 2019a). Besides, the computation overhead of GloRo training is almost free compared to other approaches so this approach is the first (and the only one at the time this thesis is published) that scales up to ImageNet-scale data and models for (deterministic) robustness certification (see our entry at a public leaderboard[1]).

The rest of the chapter is organized as follows. In Section 5.1, we introduce foundation concepts and building blocks for constructing a GloRo Net. We evaluate the robustness of GloRo Nets compared to baseline approaches in Section 5.2. We provide an empirical study on the overfitting of GloRo Nets to compare with the robust overfitting in empirical robustness discussed in Section 4.2. Section **??** uses GloRo Nets as an example to compare the locality of provable robust networks with empirically robust ones (as

[1] https://sokcertifiedrobustness.github.io/leaderboard/

discussed in Chapter 4) and our summary of this chapter is included in Section 5.3.

5.1 Construction of Globally Robust (GloRo) Nets

Certifying the robustness of a prediction can be achieved by checking if the margin between the logit of the predicted class and the others is large enough that no other class will surpass the predicted class on any neighboring point in the ϵ-ball. To this end, many prior works rely on calculating the Lipschitz Constant K (Definition 16) of the model to bound the requisite margin.

Definition 16 (K-Lipschitz Function) *A function $h : \mathbb{R}^d \to \mathbb{R}^m$ is K-Lipschitz w.r.t $S \subseteq \mathbb{R}^d$ and norm, $||\cdot||_p$, if*

$$\forall x, x' \in S \; . \; ||h(x) - h(x')||_p \leq K||x - x'||_p. \tag{5.1}$$

Namely, K is the maximum change of a function's output for changing the input in S. Notice that it is sufficient to use a *local* Lipschitz Constant K_{local} of the model, i.e. $S = \{x' | ||x' - x|| \leq \epsilon\}$ in Eq. 5.1, to certify robustness (Yang et al., 2020b). However, the local bound is often computationally expensive and needs a bounded activation to be used in training (Huang et al., 2021b). Alternatively, one can compute a *global* Lipschitz Constant K_{global}, i.e. $S = \mathbb{R}^d$ in Eq. 5.1, and leverage the relation $K_{global} \geq K_{local}$ to certify any input at any radius ϵ. A global bound is more efficient at test time for certification because it only needs to be computed once and can be used for any input at any radius ϵ.[2] However, an arbitrary global bound can be vacuously large and thus not useful for certification.

The proposed method, Globally Robust (GloRo) Nets, is a composition of global Lipschitz-bound-based certification with an effective way of regularizing the global Lipschitz Constant during training so it is tight for the certification use during the test time. The target norm in this chapter is ℓ_2 so we simply write $\|\cdot\|$ and omit p from the subscription. See an discussion of ℓ_∞ case in Section 5.3.

5.1.1 GloRo Certification

Recall that in Chapter 2 we use $f : \mathbb{R}^d \to \mathbb{R}^m$ to denote a neural network that takes an input $x \in \mathbb{R}^d$ and outputs a logit vector for m classes. The integer prediction of f is further denoted as the uppercase $F(x) = \arg\max_i f_i(x)$. GloRo computes an upper-bound of the global Lipschitz constant, K_{ji} of the logit margin between the predicted class, say $j = F(x)$, and *every other class*, i, by taking the product of the Lipschitz constant of each constituent layer as follows:

$$K_{ji} \leq ||w_j^L - w_i^L|| \cdot \prod_{l=1}^{L-1} K^{(l)}, \tag{5.2}$$

[2] Furthermore, when the network is *trained for certification with the global bound*, the global bound has the same certification potential as the local bound (Leino et al., 2021d).

where w_j^L is the j-th column of the top-layer's weight matrix and $K^{(l)}$ is the Lipschitz Constant of the l-th layer. Put together, the output of a GloRo Net follows Definition 17.

Definition 17 (GloRo Prediction) *For a GloRo Net $F^\epsilon : \mathbb{R}^d \to [m] \cup \{\bot\}$ with Lipschitz logit output $f : \mathbb{R}^d \to \mathbb{R}^m$ for class 0 to $m-1$, let*

$$j = \arg\max_{i \in [m]} f_i(x),$$

and K_{ji} be an upper-bound of the global Lispchitz constant for $f_{ji}(x) = f_j(x) - f_i(x)$ defined in Equation 5.2, F^ϵ equals to

$$F^\epsilon(x) = \begin{cases} j & \text{if } f_j(x) \geq \max_{i \neq j} \left[f_i(x) + \epsilon K_{ji} \right] \\ \bot & \text{otherwise} \end{cases}.$$

The soundness of GloRo certification can be formally shown by Theorem 6.

Theorem 6 *If GloRo Net $F^\epsilon(x) \neq \bot$, the corresponding standard model $F(x)$ is ϵ-local robust at x.*

Proof. Let $j = F^\epsilon(x)$. $j \neq \bot \implies f_j(x) \geq \max_{i \neq j}[f_i(x) + \epsilon K_{ji}]$. For any x' such that $\|x' - x\|_p \leq \epsilon$,

$$\begin{aligned} |(f_j(x') - f_i(x')) - (f_j(x) - f_i(x))| &\leq \epsilon \|x' - x\|_p \text{ (Definition 16).} \\ |(f_j(x') - f_j(x)) - (f_i(x') - f_i(x))| &\leq \epsilon K_{ji} \\ (f_j(x') - f_j(x)) - (f_i(x') - f_i(x)) &\geq -\epsilon K_{ji} \\ f_j(x') &\geq f_i(x') + f_j(x) - f_i(x) - \epsilon K_{ji} \end{aligned}$$

Because $j \neq \bot$,

$$\forall i \neq j, f_j(x) \geq f_i(x) + \epsilon K_{ji} \implies f_j(x) - f_i(x) - \epsilon K_{ji} \geq 0.$$

Thus, we arrive

$$\forall i \neq j, f_j(x') \geq f_i(x') \implies F(x') = j = F(x),$$

which shows that $F(x)$ is ϵ-locally robust at x.

Tightness of the Bound. The product of layer-wise Lipschitz constants, i.e. RHS of Equation 5.2, is concerned to be a loose bound for the function's true Lipschitz constant. To see this, let

$$r(x) = r_1(r_2(x)) \text{ and } u = \arg\max_x \frac{\|r(x)\|}{\|x\|}. \tag{5.3}$$

Algorithm 2: Power Iterations

Hyper-parameters: The number of iteration N.
Inputs: An affine transformation $A : \mathbb{R}^d \to \mathbb{R}^m$ and its transpose operation A^*.
// For matrix product, $A^* = A^\top$. For convolution, A^* is transpose convolution.
Output: Operator Norm of A.

$x \leftarrow$ random_vector();
$x \leftarrow \frac{x}{\|x\|}$;
foreach $i \in [N]$ **do**
  $x \leftarrow A^*(A(x))$;
  $x \leftarrow \frac{x}{\|x\|}$;
end
return: $\|A(x)\|$

Here $\max_x \|r(x)\|/\|x\|$ is the solution to Equation 5.1 thus the Lipschitz constant $K^{(r)}$ of u. Observe that the layer-wise upper-bound, i.e., $K^{(r_1)} \cdot K^{(r_2)}$ (where $K^{(r_i)}$ is the Lipschitz constant of r_i), is tight for $K^{(r)}$ only if u and $r(u)$ point in the *same direction* in the representation space. This is presumably unlikely to happen in a standard network and empirically demonstrated (Huster et al., 2018), as random vectors are almost orthogonal in high-dimensional space. However, what is overlooked by the prior work is that by properly regularizing this layer-wise upper-bound during the trianing, e.g. Equation 5.2, it is possible to tighten the layer-wise bound so it can be useful for certification. In Section 5.1.3, we provide greater details on GloRo traning scheme with empirical evidence that Equation 5.2 is a tight upper-bound in GloRo Nets in Section 5.2.

Computing Layer-wise Lipschitz Constant. Computing the Lipschitz upper-bound in Equation 5.2 requires the access to the Lipschitz constant of each constitute layer in the model. Our implementation uses the operator norm of an affine transformation, i.e. convolution and dense layers, as its Lipschitz constant (Gouk et al., 2021) where the operator norm is calculated efficiently with power iterations (Farnia et al., 2019; Gouk et al., 2021) (Algorithm 2). With a large number of iterations, the power method is guaranteed to converge to the operator norm of any affine transformation. To provide a precise bound for certification, we run till convergence for the power method during the test time. Although batch normalization is linear and consistent at the test time, our experiments together with other works (Huang et al., 2021b; Singla & Feizi, 2021) have not seen normalizing activation during training and testing help to improve the robustness.

Gradient-Norm Preservation (GNP) Activation. Most activation functions, e.g. ReLU, Softplus and GeLU (Hendrycks & Gimpel, 2016), are 1-Lipschitz. However, their local Lipschitz constant may not be 1 all the time, meaning the global bound can be loose for a particular set of inputs. For example, ReLU is inactivated and behaves as a constant function for negative pre-activations. For any activation

function $\alpha : \mathbb{R}^d \to \mathbb{R}^d$, $\sup_x ||\nabla_x \alpha(x)|| = 1$ is sufficient for α to be 1-Lipschitz but our goal is to have $||\nabla_x \alpha(x)|| = 1$ almost everywhere, i.e. *gradient-norm preservation* (GNP), so the 1 Lipschitz bound is tight almost everywhere. An example and popular design of GNP activation is MinMax function (Definition 18), which belongs to a family of GroupSort activations (Anil et al., 2019).

Definition 18 (MinMax (Anil et al., 2019)) *A MinMax activation* $\alpha_m : \mathbb{R}^2 \to \mathbb{R}^2$ *is equal to*

$$\alpha_m\Big(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\Big) = \begin{bmatrix} \min(x_1, x_2) \\ \max(x_1, x_2) \end{bmatrix}.$$

In short, MinMax swaps two neighboring inputs if the first is not greater than the second (or the other way around but in this thesis we use the order of min-max). This is equivalent to multiple a permutation matrix with the input. Regardless of the actual permutation, which is a function of the input, any permutation matrix has a constant singular value of 1 so MinMax is GNP and 1-Lipschitz both locally and globally. Our empirical results (to follow in Section 5.2) have shown that MinMax function is consistently better than ReLU in training provably robust models. A generalization of MinMax is HouseHolder activation (Singla et al., 2022) and is demonstrated to outperform MinMax for a particular set of networks, i.e. a network with its global Lipchitz equal to 1; however, the similar success has not been observed in GloRo Nets so we use MinMax as the standard replacement for ReLU unless noted otherwise.

5.1.2 Tightening Lipschitz Bounds in ResNet

For layers with an additional skip connection, i.e.

$$r_{\text{std}}(x) = x + g(x), \tag{5.4}$$

where g is a small network often with 2-3 convolutional layers and activation functions. Because $r_{\text{std}}(x)$ is not an affine transformation, power methods fail in this case. Instead, it is conventional to use $1 + K^g$ to be an upper-bound of the Lipschitz constant for $r_{\text{std}}(x)$ (Gouk et al., 2021). However, $1 + K^g$ is unlikely to be tight due to the same problem of $r(x)$ in Equation 5.3, even though we have the exact solution to the Lipschitz constant K^g. Our empirical results show that ResNets (He et al., 2015) using $1 + K^g$ as the layer-wise Lipschitz constants for robustness certification do not outperform feed-forward Convolutional Networks (ConvNets). This is counter-intuitive because ResNets have been found to be more effective for scaling network depth compared to ConvNets, and therefore are expected to benefit from larger network capacity to learn more robust decision boundaries.

LiResNet Layer. An alternative to the conventional ResNet layer, i.e. Equation 5.4, which contains a skip connection but has a much tighter Lipschitz constant other than $1 + K^g$ is to inject the skip connection

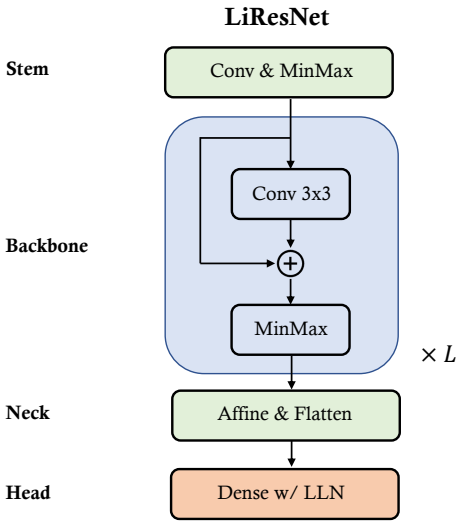


Figure 5.2: LiResNet Architecture. We additionally use Last Layer Normalization (LLN) (Singla et al., 2022) in the Head. Architecture details are included in Appendix C.

inside a convolutional layer, as proposed in our recent work (Hu et al., 2023). That is, we define

$$r_{\text{linear}}(x) = x + \texttt{conv}(x).$$

Because $r_{\texttt{linear}}(x)$ is an affine transformation w.r.t x, its Lipschitz constant of $r_{\texttt{linear}}$ can be tightly computed by directly applying power methods, which is always smaller or equal to the addition-form bound (i.e., $1 + K^{\texttt{conv}}$). By stacking multiple linear residual blocks (with interjecting activations), and including a stem, neck, and head, we obtain the *Linear ResNet* (LiResNet) architecture, illustrated in Figure 5.2 with more details in Appendix C.

In the evaluation sections, we show LiResNets outperform the conventional ResNets and ConvNets on various data distributions, proving its unique role in learning certifiable robust models.

5.1.3 GloRo Training

A straightforward way to encourage the model to have certified prediction is penalizing its Lipschitz constant so the margins between the top prediction and other candidates shrink less when the input is perturbed. Intuitively, one can regularize the Lipschitz constants with a penalty coefficient along with the standard loss. However, it is unclear how one should set or schedule the penalty coefficient to balance between learning a smooth but expressive network. Over-regularizing the Lipzchitz constant may lead towards a degenerated model, whereas under-regularization fails to certify the prediction. To address the

hardness of using a regularization term to control Lipschitz constants, which is inefficient and tricky in practice, we provide the following two techniques.

Training with an Extra $\perp$ Class. Notice that during the test time, a GloRo Net abstain by outputting $\perp$ for points that can not be certified. During training, we can construct an artificial logit score for the $\perp$ class, denoted as $f_\perp(x)$, and directly use conventionally loss function, e.g. Cross-Entropy loss (CE), to reward the network for not letting $f_\perp(x)$ surpass the logit score of the true label. Based on our certification approach (Definition 17), $f_\perp(x)$ needs to be at least the logit score of the most competitive class plus the Lipschits constant of the margin. Namely, suppose $j = \arg\max_i f_i(x)$, we set

$$f_\perp(x) = \max_{i \neq j} \left[f_i(x) + \epsilon K_{ji} \right].$$

Because no natural input is labeled as $\perp$, the network must learn to decrease $f_\perp(x)$ in order for the logit of a true label to surpass $f_\perp(x)$. On the other hand, it is not necessary for the model to have $f_\perp(x) = 0$ because it is sufficient to have $f_j(x) \geq f_\perp(x)$ to certify the prediction so further decreasing $f_\perp(x)$ at x will not be rewarded as much as getting another point correct. Adding an extra logit score helps to provide gradient directions that precisely do what we desire for certifiable robustness and avoid abusing the capacity of the network from learning over-regularized models. An illustration of the $\perp$ logit can be seen in Figure 5.3.

With a logit vector $[f_0(x), \cdots, f_{m-1}(x), f_\perp(x)]$ for an input x, the training objective uses a standard Cross Entropy (CE) or a loss inspired by TRADES (Zhang et al., 2019b) (previously discussed in Chapter 4 with Definition 15) defined as follows.

Definition 19 (GloRo-CE Training) *Given a GloRo Net F^ϵ and the $\perp$ logit score $f_\perp(x)$, GloRo-CE loss for (x, y) equals to*

$$\mathrm{CE}((x, y), f^\epsilon(x)),$$

where $f^\epsilon(x) = [f_0(x), \cdots, f_{m-1}(x), f_\perp(x)]$.

Definition 20 (GloRo-TRADES Training) *Given a GloRo Net F^ϵ and the $\perp$ logit score $f_\perp(x)$, GloRo-TRADES loss for (x, y) equals to*

$$\mathrm{CE}((x, y), f) + \lambda_t \cdot \mathrm{kl}(\mathrm{softmax}(f^\epsilon(x)) || \mathrm{softmax}(f(x))),$$

where $f^\epsilon(x) = [f_0(x), \cdots, f_{m-1}(x), f_\perp(x)]$ and λ_t is a hyper-parameter.

Figure 5.3: Illustrations of GloRo-CE (Definition 19), GloRo-TRADES (Definition 20) and GloRo-EMMA (Definition 21) losses.

Efficient Margin MAximization (EMMA). One potential issue for using the $\perp$ class is it can be suboptimal at penalizing all logit margins that are not large enough. That is, when there are multiple threatening candidate classes which are all close to the top one, the gradient update from GloRo-CE or GloRo-TRADES only penalizes the most threatening class. Ultimately, this need not be a problem—by minimizing the loss originating from the *worst-case* adversary, the network successfully guards against *any* possible adversarial example. However, at each iteration, only one competing class faces a strong negative gradient signal even if the threatening class is only slightly more competitive than the others. For example, in Figure 5.3, GloRo-CE/TRADES training at iteration t largely focuses on the decision boundary between class 1 and 3 even though class 2 is also competitive but slightly less so than class 3. Furthermore, the possibility arises, then, that the threatening class will alternate between competing classes in a sort of "whack-a-mole" during training.

Indeed, we find this phenomenon to be increasingly common as the number of classes grows. Figure 5.4 shows the fraction of instances at each epoch for which the threatening class differs from in the preceding epoch. In the initial 100 epochs, more than 30% instances from Tiny-ImageNet (containing 200 classes) have different threatening classes at each each iteration, while the same number for CIFAR-10 is only about 10%. At best, this contributes to making training with many classes less efficient, and at worst, it halts progress as work in one epoch is undone by the next.

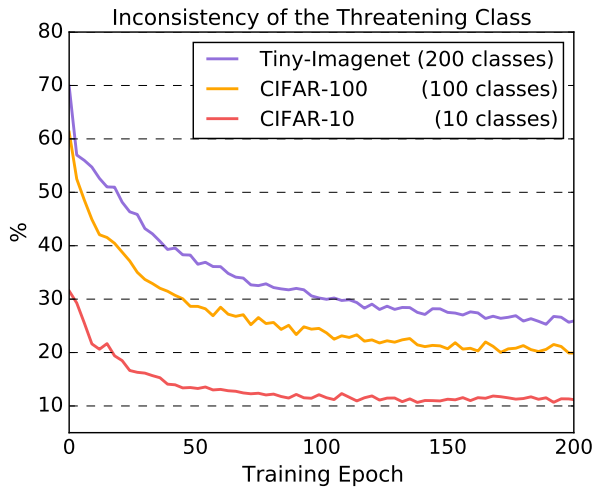


Figure 5.4: Plot of the percentage of instances at each epoch during training for which the *threatening* class (the one with the second-highest logit value) differs compared to the previous epoch (on the same instance). Numbers are reported on three datasets with 10, 100, and 200 classes using a GloRo LiResNet with TRADES loss.

To address this problem, we propose *Efficient Margin MAximization* (EMMA) loss (Definition 21) for training GloRo Nets. EMMA loss can be conceptualized as adding to each non-ground-truth class the maximum margin (ϵK_{yi}), which the respective logit could gain on the ground truth logit within an ϵ neighborhood (illustrated in Figure 5.3). Effectively, this enables us to penalize adversarial examples from *all* classes simultaneously (as illustrated in Figure 5.3), leading to a less sparse gradient signal.

Definition 21 (Efficient Margin Maximization loss) *Given a GloRo Net F^{ϵ} with the logit output f, GloRo-EMMA loss for data (x, y) equals to*

$$-\log \frac{\exp\left(f_y(x)\right)}{\sum_i \exp\left[f_i(x) + \tilde{\epsilon}_i K_{yi}\right]} \text{ and } \tilde{\epsilon}_i = clip\_value\_no\_grad(\kappa_i(x), 0, \epsilon)$$

When using EMMA loss, the actual radius at which we require the model to certify robustness at each iteration is $\tilde{\epsilon}_i$ for the margin between class i and y. It's important to note that if the model still predicts class i over the ground truth class y (i.e., $f_y(x) < f_i(x)$ thus $\kappa_i(x) < 0$), we clip $\tilde{\epsilon}_i$ to 0 and EMMA loss reduces to a standard Cross Entropy loss (for class i). In this case, the training focuses on improving clean accuracy first. As the training progresses, $\tilde{\epsilon}_i$ may grow to the same magnitude as the expected radius ϵ. Once this happens, we shift the model's focus towards certifying other instances, rather than continuously increasing the robust radius for x. We need to stop the gradient back propagation from $\tilde{\epsilon}_i$ and treat it as a number. Otherwise the denominator term $f_i(x) + \tilde{\epsilon}_i K_{yi}$ is just $f_y(x)$ and the logit for class i is removed from the computation graph of the loss.

EMMA may bear similarities to Lipschitz-margin training (LMT) (Tsuzuku et al., 2018), which adds $\sqrt{2}\epsilon K$ to all non-groundtruth logits and K is the global Lipschitz constant of F. However, here are two fundamental differences that distinguish EMMA. First, instead of employing the square-root bound, EMMA directly utilizes the Lipschitz constant for each margin, i.e. K_{yi}, which provides a tighter bound. Second, the robust radius used in EMMA is $\tilde{\epsilon}$ instead of the radius ϵ used in testing. By the definition of EMMA, $0 \leq \tilde{\epsilon} \leq \epsilon$ and $\tilde{\epsilon}$ is 0 if the model is not predicting the input correct. $\tilde{\epsilon}$ grows as the model becomes more robust at the corresponding input. As a result, when the model is not sufficiently robust at the input, EMMA uses $\tilde{\epsilon} < \epsilon$ and imposes a milder robust regularization. On the other hand, LMT always adds $\sqrt{2}\epsilon K$ to all non-groundtruth logits even before the model is capable of predicting the label correct.

The dynamic margin used in EMMA loss is important. If using a fixed margin, the loss function turns out to be:

$$\ell_{\text{fixed}} = -\log \frac{f_y(x)}{\sum_i f_i(x) + \epsilon K_{yi}}$$

The fixed margin loss ℓ_{Fixed} penalties the margin Lipschitz between the ground truth class and all other classes. Therefore, this loss function imposes a stronger regularization on the Lipschitz constant of the model than EMMA loss, and limits the model capacity more. We find that models trained with the fixed margin loss require weaker data augmentation or smaller training ϵ to avoid underfitting. However, this will make the model robust overfitting. The gap between validation clean accuracy and validation VRA is increased for the fixed margin loss and the validation VRA is lower than models trained with the dynamic margin loss, i.e., EMMA loss.

Number of Power Iterations in Training. To certify the prediction in the test time, power iterations need to run till convergence for precise results. However, during training, the bound is not necessary to be as precise as the test time. What matters more is that we need the gradient signal from power methods to provide a direction so we can efficiently decrease the operator norm. In practice, we find setting $N = 5$ or 10 is sufficient for the purpose of training. For readers who are interested in the effect of the number of power iterators on the gradient update direction, we refer to Leino (2022).

5.2 Evaluations

5.2.1 Evaluating Robustness

This section evaluates the provable robustness of GloRo Nets compared to other baselines. Different from using empirical attacks for evaluating robustness in Section 4.5, we focus on *Verifiable Robust Accuracy* (VRA), i.e. the percentage of points that are both correct and on which the model is certifiably robust, as our metric

in this section. We assume all adversaries are ℓ_2-norm-bounded with a radius ϵ. For MNIST, we report VRA at $\epsilon = 1.58$. For CIFAR-10/100, Tiny-ImageNet and ImageNet, we report VRA at $\epsilon = 0.141 \approx 36/255$. Our use of ϵ for each dataset follow the standard choices in the literature (Araujo et al., 2023; Huang et al., 2021b; Lee et al., 2020a; Meunier et al., 2022; Singla & Feizi, 2021; Singla et al., 2022; Wong & Kolter, 2018a,b). For baseline methods, we use the best VRAs reported by their original papers.

Baselines. On MNIST, the baseline provably robust networks include KW (Wong & Kolter, 2018b) and BCP (Lee et al., 2020a). On CIFAR-10/100, KW and BCP are no longer competitive baselines after the publication of our initial GloRo paper (Leino et al., 2021b) so they are not included for these datasets. We instead include BCOP (Li et al., 2019b), Cayley (Trockman & Kolter, 2021), Local-Lip (Huang et al., 2021b), SOC(HH+CR) (Singla et al., 2022) and LOT (Xu et al., 2022), CPL (Meunier et al., 2022), which are published after the initial GloRo paper (Leino et al., 2021b) but earlier than our follow-up work (Hu et al., 2023). Additionally, we include SLL (Araujo et al., 2023), a concurrent work to (Hu et al., 2023). On Tiny-ImageNet, we include Local-Lip (Huang et al., 2021b) and SLL (Araujo et al., 2023) because other baselineds do not run experiments on this dataset. For ImageNet, we are the only and the first one to report VRA at the time this thesis is written. For readers who are interested in the comparison of our results to older or newer methods, please find our entries on the publick leaderboard[3].

Network Architecture. We build GloRo Nets using ConvNets and LiResNets. For ConvNets, we follow the layer specifications used in prior work (Wong & Kolter, 2018b). For example, we use 6C2F to denote a ConvNet with 6 convolutional layers and 2 fully-connected (dense) layer. For LiResNet, we a model with 6 linear residual layer and 128 channels for each convolutional module inside the residual layer as L6128W. We use MinMax as the activation function for all architectures. More architecture details can be found in Appendi **??**. Network architectures used in prior works, if different from our choices, are denoted in the same way as their original papers do. To quickly distinguish the size of one architecture from another, we map the number of network parameters from integer values to the following labels.

- the number parameters $< 5M \implies$ S.
- the number parameters from $5M$ to $10M \implies$ M.
- the number parameters from $10M$ to $50M \implies$ L.
- the number parameters >$50M \implies$ XL.

[3] https://sokcertifiedrobustness.github.io/leaderboard/

method	Arch.	Config.	Size	`Clean` (%)	`ERA` (%)	`VRA` (%)
MNIST ($\epsilon = 1.58, SE = \pm 0.5\%$)						
KW (Wong & Kolter, 2018b)	ConvNet	4C3F	S	88.1	67.9	44.5
BCP (Lee et al., 2020a)	ConvNet	4C3F	S	92.4	65.8	47.9
Gloro Net (TRADES)	ConvNet	4C3F	S	97.0	81.9	**62.8**
CIFAR-10 ($\epsilon = 0.141, SE = \pm 0.5\%$)						
BCOP (Li et al., 2019b)	ConvNet	6C2F	S	75.1	67.3	58.3
Cayley (Trockman & Kolter, 2021)	ConvNet	6C2F	S	75.3	67.7	59.2
Local-Lip Net (Huang et al., 2021b)	ConvNet	6C2F	S	77.4	70.4	60.7
SOC (HH+CR) (Singla et al., 2022)	ConvNet	LipConv-15	M	76.4	-	63.0
LOT (Xu et al., 2022)	ConvNet	LipConv-25	M	77.1	-	64.3
CPL (Meunier et al., 2022)	ResNet	70C15F	XL	78.5	-	64.4
SLL (Araujo et al., 2023)	ResNet	120C15F	XL	73.3	-	<u>65.8</u>
Gloro Net (TRADES)	ConvNet	6C2F	S	77.0	69.2	60.0
Gloro Net (EMMA)	LiResNet	L6W128	S	76.5	70.6	63.7
Gloro Net (EMMA)	LiResNet	L18W256	M	78.1	72.0	**66.0**
CIFAR-100 ($\epsilon = 0.141, SE = \pm 0.5\%$)						
BCOP (Li et al., 2019b)	ConvNet	LipConv-10	M	45.4	-	31.7
Cayley (Trockman & Kolter, 2021)	ConvNet	LipConv-10	M	45.8	-	31.9
SOC (HH+CR) (Singla et al., 2022)	ConvNet	LipConv-20	L	47.8	-	34.8
LOT (Xu et al., 2022)	ConvNet	LipConv-30	L	49.2	-	35.5
CPL (Meunier et al., 2022)	ResNet	70C15F	XL	47.8	-	33.4
SLL (Araujo et al., 2023)	ResNet	120C15F	XL	46.5	-	36.5
Gloro Net (EMMA)	LiResNet	L6W128	S	49.6	43.4	35.4
Gloro Net (EMMA)	LiResNet	L18W256	M	51.2	44.7	**38.3**
Tiny-ImageNet ($\epsilon = 0.141, SE = \pm 0.5\%$)						
Local-Lip Net (Huang et al., 2021b)	ConvNet	8C2F	M	36.9	33.3	23.4
SLL (Araujo et al., 2023)	ResNet	120C15F	XL	32.1	-	23.2
Gloro Net (EMMA)	LiResNet	L6W128	S	37.8	33.8	27.0
Gloro Net (EMMA)	LiResNet	L18W256	M	40.7	36.3	**29.2**
ImageNet ($\epsilon = 0.141$)						
Gloro Net (EMMA)	LiResNet	L18W588	M	45.6	-	**35.0**

Table 5.1: Comparing GloRo Nets with other provably robust networks on *Verifiably Robust Accuracy* (VRA), i.e the percentage of test points that are both accurate and their predictions are locally robust. A max Standard Error (SE) (Stark & University of California, 2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (n as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

Main Results. In Table 5.1, the top results are highlighted in bold font. To measure the significance of an improvement, we calculate the maximum standard error (SE) (Stark & University of California, 2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (where n denotes the number of test examples) for each dataset. Thus, the accuracy of a certain model is regarded a *silver* result a_{silver}, if its SE interval overlaps with the that of the top result a_{top}. Namely, $a_{top} - SE \leq a_{silver} + SE$.

On MNIST, GloRo Nets outperform the previous best results by around 15% wit the same network, showing the promise of GloRo Nets of utilizing the network capacity. On CIFAR-10/100, the GloRo Nets with LiResNet and EMMA loss are better than all baseline results. Notice that our LiResNets are

Architecture	dataset	TRADES	EMMA
ConvNet	CIFAR-10	58.8	59.2
	CIFAR-100	34.0	35.0
	Tiny-ImageNet	26.6	27.4
LiResNet	CIFAR-10	66.2	66.3
	CIFAR-100	37.3	37.8
	Tiny-ImageNet	28.8	30.3

(a)

Dataset	L	ConvNet	ResNet	LiResNet
CIFAR-10	6	64.0	60.3	65.5
	12	59.2	60.0	66.3
	18	×	60.1	66.6
CIFAR-100	6	36.5	33.5	37.2
	12	35.0	33.5	37.8
	18	×	33.6	38.0

(b)

Table 5.2: (a) VRA performance (%) of a ConvNet and a LiResNet on three datasets with different loss functions. (b) VRA (%) performance on CIFAR-10/100 with different architectures (L is the number of blocks in the model backbone). We use EMMA loss for Gloro training. All models in this table use 256 channels in the backbone. A value of × indicates that training was unable to converge.

much smaller than the architecture used by SSL with approximately 0.15x layers than the model used by SSL (Araujo et al., 2023). On Tiny-ImageNet, GloRo Nets are better than two baselines and our best result, i.e. LiResNetand EMMA loss, is better than the previous state-of-the-art by 5.8%. Interestingly, we see that with a much smaller network, GloRo Net is much better than a much larger network trained with SSL. This is even obvious when comparing GloRo Nets with a network size of M, e.g. L6128W, with SSL using a network size of XL. Our results on Tiny-ImageNet demonstrate the effectiveness of GloRo Nets on leveraging the capacity of neural nets to learn robust decision boundaries.

There is no theoretical limitation of the baseline approaches that would prevent us from directly training them on ImageNet; however, practical resource constraints prevent us from training and wait until convergence. For example, baselines using orthogonalized kernels—e.g., Cayley, BCOP, and SOC—do not easily fit into memory with $224 \times 224 \times 3$ images, and local Lipschitz computation—e.g.,Local-Lip Net—is both time and memory intensive. Thus, to the best of our knowledge, we are the first to report the VRA on ImageNet with a *deterministic* robustness guarantee.

5.2.2 Ablation Study for Architecture and Loss

We mainly focus on LiResNet and EMMA loss in Table 5.1 because we found they have surpassed other configurations. We run an ablation study on the choice of architecture and another on the choice of loss. Results are shown in Table 5.2.

For both ConvNets and LiResNets, we experiment on CIFAR-10, CIFAR-100 and Tiny-ImageNet and report VRAs in Table 5.2a. ConvNets are modified to have same channel size (i.e. wider) as LiResNets and we end up achieving higher VRAs than Leino et al. (2021d). For all LiResNets, we adjust the configuration of the first convolution in the Tiny-ImageNet models to have the same output size compared to the CIFAR models, minimizing the impact of the image size on our conclusions (as the key variable in these experiments is the *number of classes*). The performance gain from switching CE or TRADES to EMMA on

GloRo Nets becomes clear on Tiny-ImageNet than that is on CIFAR-10. When noticing that Tiny-ImageNet has 200 classes while CIFAR-10 only has 10 classes, this observation aligns with our hypothesis used to motivate EMMA loss discussed in Section 5.1.3. Namely, the rotating threatening class phenomenon in GloRo-CE/TRADES may contribute to suboptimal learning.

Table 5.2b includes another ablation study for the choice of architecture. We run head-to-head comparisons on CIFAR-10 and CIFAR-100 of GloRo Nets using (1) a feed-forward ConvNet architecture, (2) a conventional ResNet architecture, and (3) a LiResNet architecture. We train all three architectures with EMMA loss at three different depths. We see in Table 5.2b that very deep ConvNets may not be able to converge even on small-scale datasets like CIFAR-10 and CIFAR-100. Moreover, the VRA of both ConvNets and conventional ResNets do not benefit from the increasing network depth – in fact, performance *decreases* as the network is made significantly deeper. The observed performance decrease and convergence issue on ConvNets are perhaps not specific to robust training as this is one of the acknowledged reasons why we switch from ConvNets to ResNets in standard training—very deep ConvNets suffer from gradients issues. However, in the case of certified training with very deep ConvNets, this problem may be amplified since the training objective imposes strong regularization from the Lipschitz estimation. On the other hand, we see that ResNet and LiResNet do not suffer from the same convergence issues because of the use of skip connections. However, in comparison to ResNet, LiResNet is the only architecture under the same conditions that benefit from more layers, as the over-estimation of the Lipschitz constant in ResNets results in overregularization, wasting the model's capacity.

5.2.3 Evaluating the Tightness of the Bound

Weng et al. (2018a) report that a layer-wise product of Lipschitz upper-bound on the global Lipschitz constant is not capable of certifying robustness for a non-trivial radius. While this is true of models produced via *standard training*, GloRo Nets impose a strong implicit regularization on the global Lipschitz constant. Indeed, Table 5.3 shows that the global upper bound is several orders of magnitude smaller on GloRo Nets than on standard networks.

Another potential limitation of using an upper bound of the global Lipschitz constant is the bound itself (Huster et al., 2018). Table 5.3 shows that a lower bound of the Global Lipschitz constant, obtained via running gradient descend using Definition 16 (see details in Appendix C), reaches an impressive 83% of the upper bound on MNIST, meaning that the upper bound is fairly tight. On CIFAR-10 and Tiny-Imagenet the lower bound reaches approximately 70% and 47% of the upper bound, respectively. However, on a standard model, the lower bound is potentially orders of magnitude looser. These results suggests the

method	global UB	global LB	local LB
MNIST ($\epsilon = 1.58$)			
Standard	$5.4 \cdot 10^4$	$1.4 \cdot 10^2$	17.1
GloRo Net	2.3	1.9	0.8
CIFAR-10 ($\epsilon = 0.141$)			
Standard	$1.2 \cdot 10^7$	$1.1 \cdot 10^3$	96.2
GloRo Net	15.8	11.0	3.7
Tiny-Imagenet ($\epsilon = 0.141$)			
Standard	$2.2 \cdot 10^7$	$3.6 \cdot 10^2$	40.7
GloRo Net	12.5	5.9	0.8

Table 5.3: Comparing the tightness of global Lipschitz bound using layer-wise product on standard and on Gloro Nets.

objective imposed by the GloRo Net helps by incentivizing parameters for which the upper bound estimate is sufficiently tight for verification.

Finally, we compare the global upper bound to an empirical lower bound of the local Lipschitz constant. The local lower bound given in Table 5.3 reports the *mean* local Lipschitz constant found via optimization in the ϵ-balls centered at each of the test points. The ratio of the local lower bound to the global upper bound is essentially zero in the standard models, compared to 6-35% in the GloRo Nets, establishing that the upper bound is again much tighter for GloRo Nets. Still, this suggests that a reasonably tight estimate of the local bound may yet help improve the VRA of a GloRo Net at runtime, as is shown by Huang et al. (Huang et al., 2021b). However, the gain of VRA is less than 1% on CIFAR-10, demonstrating the sufficiency of using a global bound to certify robustness.

5.3 Related Work and Discussion

Randomized Smoothing. Contradictory to the deterministic robustness guarantee focused in this paper, probabilistic guarantees, mostly based on *Raondmized Smoothing* (RS, Definition 13) (Cohen et al., 2019a), have been long studied and experimented on ImageNet-scale models, motivating a set of smoothing-aware training approaches to further increase the robust radius (Carlini et al., 2022; Jeong et al., 2021; Salman et al., 2019, 2020). On ImageNet, RS indeed reports certifying more points than Gloro LiResNet at the same ϵ. However, the fundamental limitation of a probabilistic guarantee is the false positive results, meaning that adversarial examples can go unchecked. Even a 0.1% false positive rate (FPR) is not affordable to many real-world security and financial applications. On the other hand, RS-based certification has order-of-magnitude larger computation overhead compared to Lipschitz-based certification. For the same example of FPR=0.1%, RS runs 100,000 extra inferences to certify 1 instance and the number grows up

exponentially to achieve lower FPR (Cohen et al., 2019a). This resource-consuming nature of RS-based certification in the end limits the type of applications one can deploy it for.

ℓ_∞ Certification. While in this work, we focus on the ℓ_2 norm, the ideas presented in Chapter 5 can be applied to other norms, including the ℓ_∞ norm. However, we find that the analogue of the approximation of the global Lipschitz bound given Equation 5.2 is loose in ℓ_∞ space. Meanwhile, a large volume of prior work applies ℓ_∞-specific certification strategies that proven effective for ℓ_∞ certification (Balunovic & Vechev, 2020; Zhang et al., 2020).

Tightening Lipschitz Bound with Better Architecture. The closest line of work on improving VRAs with architecture redesign is the use of othornormalized convolution (Singla & Feizi, 2021; Trockman & Kolter, 2021), which is *exactly* 1-Lipschitz by construction. In a similar spirit, we introduce the linear residual branch in LiResNet to solve the overestimation of Lipschitz Constant in the conventional ResNet. Our linear residual layer compares favorably to othornormalized layers in a few key advantages. The linear branch is open to any affine transformation with no restriction on its kernel. Although there is no technical limitation that would prevent us from using othornormalized convolution in LiResNet, a standard convolution works favorably well in our experiments and is significantly less expensive than training with othornormalized ones. As is shown in Table 5.1, it is worth mentioning that all prior works which attempt to scale up feed-forward ConvNets is less efficient than the LiResNetproposed for GloRo Nets, whereas the skip connection allows us to effectively utilize the increasing capacity and improve VRAs on various datasets.

Robustness Needs More Capacity. Despite the fact that robust and correct predictions, i.e., VRA=100%, is achievable for many standard datasets (Yang et al., 2020b), the state-of-the-art VRAs (which are achieved using Lipschitz-based certification) are far away from realizing this goal. Besides the fact Lipschitz-based certification is not exact and may falsely flag robust points, recent works have emphasized the role of network *capacity* in learning robust classifiers. Bubeck (Bubeck & Sellke, 2021) showed that a smooth (and thus robust) decision boundary requires d times more parameters than learning a non-smooth one, where d is the ambient data dimension. In addition, to *tightly certify* a robust boundary with Lipschitz-based approaches, Leino (Leino, 2023) demonstrates the need for extra capacity to learn smooth level curves around the decision boundary, which are shown to be necessary for tight certification.

Fundamental Limitation in Robust Classification. Our motivation on EMMA and the empirical observations on the impact of class numbers to VRAs expose the hardness in scaling up robustness certification to

real-world datasets. Training to be both robust and accurate has to balance the rewards between predicting the groundtruth and pushing away threatening boundaries, which often becomes precarious and tricky as the classes increase. In addition, the problem is potentially compounded by ambiguous or conflicting class labels, which may be indicative of a mismatch between the data distribution and the learning objective of *categorical accuracy*—and particularly its robust form. A number of mislabeled images have been identified in ImageNet (Beyer et al., 2020; Northcutt et al., 2021; Vasudevan et al., 2022; Yun et al., 2021), placing difficulties for robust classification. Moreover, real-word images are often not well-cropped input that only contains one semantically meaningful object. Robustly assigning only one label to an image with multiple objects, the often case in ImageNet, again limits the function networks can learn – the network either learn to not see the rest of objects or find that there is no way to be certifiably robust. In such cases, robust learning may be essentially limited unless alternate objectives (e.g., robust analogs of top-k accuracy (Leino & Fredrikson, 2021) and robust segmentation (Fischer et al., 2021)) are pursued.

Chapter 6

Conclusions And Future Work

In this last chapter, we first summarize our approaches, results, and achievements from the previous chapters in Section 6.1. Our work on improving locality by seeking for better tools to improve the robustness is an important step but probably far from sufficient towards promoting feature alignment. In the rest of this chapter, we discuss possible limitations of locality and robustness in realizing well-aligned models, in both vision and non-vision tasks uncovered by our previous discussion. We propose a set of future works to address issues that cannot by directly solved by robustness, together with a set of preliminary results as a proof of concept.

6.1 Conclusions

In this thesis, we concentrate on one of the fundamental requirements for deep neural networks to align with humans' interests. That is, when they are used as image classifiers, their outputs are expected to be always equivalent to human experts' answers, which we term as *output alignment*. Given that it is not feasible to test every sample from the distribution, we instead consider an essential property for the model to generate aligned outputs — the important features that contribute to the model's output must be semantically proper to the task from a human's perspective. For the domain, i.e. image classification, studied in this thesis, we say a model is *aligned* with humans on the use of features if influential pixels to the model's output class are the ones that humans would agree and find relevant to the label.

To quantitatively measure the feature alignment, we need tools to locate important features to the model and truly important features from humans' perspective, in addition to a metric to compare their similarity. Feature attribution is a set of faithful tools that help to quantify the importance score of each input towards the model's decision at an input x. We compare these important features with the bounding box of object in

an image because it includes a human-labeled region that is associated with the label. That is, we consider input features in the bounding box is truly important. Similar to precision, recall and the F1-score, which are used to evaluate the quality of localization, we propose Pre, Rec and F1 *locality* scores for evaluating the alignment between attributions and the bounding boxes. That is, a higher geometrical mean, i.e. F1, of Pre and Rec indicates that attributions better align with the bounding box so the underlying model is aligned more on features with humans.

With the attribution-based locality metric(s), the thesis demonstrates that the Rec-locality scores of state-of-the-art deep classifiers on ImageNet, e.g.Vision Transformers (ViT) and ResNets, are not significantly improved compared to the outdated VGG nets and SqueezeNets and a random network. To improve the locality, the thesis shows the promise of *adversarial robustness*, which refers to the resistance of a model's prediction against imperceptible and adversarial noise. Our experiments on ImageNet show that improved robustness leads to improved locality over all tested models. Furthermore, more robust models have higher locality than the less robust ones, demonstrating that robustness is necessary for realizing feature alignment for classification tasks.

Our evaluation shows that improving robustness is an effective way to improve the locality of the model compared to only focusing on the accuracy of benign examples. Thus, the rest of the thesis focuses on techniques that can further improve the existing achievements on training robust models.

Recent work in training robust models is often blocked by the phenomena of *robust overfitting*, which entails a generalization gap between the test robustness and the training robustness. In particular, such gap is often larger than the generalization gap between the performance of the underlying model on benign images. In principle, there is no limitation that would prevent one from learning a robust and well-generalized function, as evidenced by the fact that humans are well-generalized and robust. In learning theory, Probably Approximately Correct-Bayesian (PAC-Bayesian) theory provides a way to characterize the test performance of any classifier. Our work therefore aims to directly minimize the PAC-Bayesian generalization bound for robustness, while prior works have skipped the minimization step and directly used the initial and loose bound. We demonstrate a practical use of our minimized PAC-Bayesian bound for alleviating robust overfitting, which includes a new TrH *Regularization* term for the top layer of the network during the training. Our empirical evaluations on CIFAR-10, CIFAR-100, and ImageNet with ViTs create new state-of-the-art robust accuracy under both ℓ_2- and ℓ_∞-norm-bounded adversaries.

However, care must be taken that empirical evaluation for robustness does not guarantee that the model is indeed robust because there could be attacks that adapt to the defense to break the defense in the future. In network verification, certifying the robustness requires a lot more resources than empirical attacks and can only finish for small architectures. In this thesis, we introduce a novel set of networks,

GloRo Nets, which are built with an *instant* robustness certification layer based on the global Lipschitz constant of the model and thus have the potential to certify much deeper architectures. GloRo Nets output a new class, $\bot$, that can be interpreted as abstention from inference if robustness at this point can not be certified and a regular class integer otherwise. Notice that our certification is deterministic so there will be not false positive case. The benefit of using a global Lipschitz constant is that this is a one-time computation for all inputs; therefore, the certification cost during the test is as fast as doing standard inference — namely, the certification is instant. Focusing on ℓ_2-norm-bounded adversaries, GloRo Nets have shown the state-of-the-art *provable* robustness on CIFAR-10, CIFAR-100, Tiny-ImageNet, and ImageNet with a deterministic guarantee.

6.2 Limitation of Robustness

While robustness is a necessary step to ensure aligned perception with humans, it is not sufficient to address all issues. One limitation considered in this thesis is an out-of-distribution (OOD) input. A specific use of OOD usually refers to points sampled from a shifted distribution to the training distribution, whereas here we use OOD to generally refer to points that the model has not seen before. So an OOD point can be an adversarial point or a point does not fall into any of the classes. Consider we are given a robust binary classifier that can tell the difference between a cat and a dog. The model was built with a robustness guarantee to use features that align better with human perceptions to differentiate between the two classes. However, if the model is given an image consisting of random noise that does not have any semantically meaningful features, it would be considered an OOD input. Nonetheless, the model could still classify this OOD input into either cat or dog robustly if the input is sufficiently far away from the decision boundary. In addition, the robust binary classifier may also classify other images that are similar but not exactly the same as the training examples into existing classes, such as a wolf that is similar to but not a dog. While these classifications may not be desired for the classifier, it is still possible for the model to make a prediction with high confidence based on the available features. Figure 6.1 illustrates this concern with more examples.

The problem of (robust) classification on out-of-distribution (OOD) inputs arises from the task being under-specified. Specifically, we have only provided a specification for the network to (robustly) classify the in-distribution points, such as cats and dogs in our previous examples, but not for OOD points. Geometrically, the decision boundary extends to infinity in the input space and over-extrapolates the function learned from in-distribution samples provided during the training. This issue cannot be completely resolved by enforcing robustness since a non-closing decision boundary can also be present in a robust

Figure 6.1: Plot of a limitation of robustness for predicting out-of-distribution inputs, which is not aligned with what humans would respond to the input, e.g. abstention. The nature of this issue is that deep models over-extrapolate (outside the dotted circle in blue) the function learned from in-distribution points to the infinity so the decision boundary is not closed.

model as illustrated in Figure 6.1.

6.3 Future Work: Selective Classification

A related line of work that has potential to address the non-closing decision boundary is *selective classification*, which often consists of a classification network $f : \mathbb{R}^d \to \mathbb{R}^m$ and a rejection network $g : \mathbb{R}^d \to \{0, 1\}$. The output label of a selective classifier is $f(x)$ if $g(x) > 0$ and $\perp$ otherwise. That is, $g(x)$ determines whether we should reject the current input and abstain the prediction by outputting $\perp$.

Prior works on determining the portion of points that should be rejected (i.e. $g(x) = 0$) is mainly based on estimating the uncertainty in the prediction of $f(x)$ (Dusenberry et al., 2020; Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Maddox et al., 2019). Recent works decide to set a threshold for the percentage of points that the model should reject in the training stage and maximize the model's performance on the rest accepted data (Feng et al., 2023; Geifman & El-Yaniv, 2019; Huang et al., 2020; Ziyin et al., 2019). These approaches often involve a stand-alone g, which is trained together with f.

However, as experiments have shown that the model's prediction on adversarial points can be confident (Madry et al., 2018a), it is unclear whether the prior works will result in a robust classifier f and a robust rejection model g. The consequence of non-robustness has been shown in detail in previous chapters – they are not well-aligned with humans' perception. Thus, for selective classification to be useful to address

the uncovered limitation of robustness, it must be robust to begin with.

Comparing to OOD Detection. Another related line of work that has a close functionality with g is to train an OOD detector and stack it with the classification model f. Many OOD detection algorithms are offline and independent of the training algorithms and the classifier f. As a result, the objectives between the OOD detector and the classifier f can be mis-calibrated. While the detector aims to clean the data and reduce the unnecessary variance in the sample set, the classifier could have the capacity to adapt the complex distribution in the data but is not given the chance. Therefore, an OOD detector that adapts to the classifier and only rejects the points that the current classifier is not capable of predicting is more proper and effective use of the capacity of deep nets.

Unlike existing works that determine the rejection using another network g, GloRo Nets introduced in Chapter 5 is an end-to-end design where g is by construction if the output logit of the $\bot$ class is higher than the top logit output. However, as we point out the limitation of robustness in learning closed decision boundary, the rejection rule of GloRo Nets needs to be more general for invalid points with non-robust reasons, i.e. being out of the classes the model has seen before. The next section introduces a new research direction, which we will term as Centroid Net with rejection. The advantage of Centroid Nets over conventional classifiers and GloRo Nets is a unified way to only predict for in-distribution points and reject out-of-distribution (i.e. invalid) points. We begin with a standard Centroid Net without rejection in Section 6.3.1 and later introduce the rejection rule in Section 6.3.2. As a proof of concept, we provide a set of analytical results and preliminary empirical evidence of justify the promise of studying and improving Centroid Nets for better alignment.

6.3.1 Standard Centroid Nets

The core component of a standard Centroid Net is a Centroid layer, which is essentially a parametric version of an 1-Nearest Neighbor (1-NN) classification model. Recall that the conventional 1-NN model assigns the label of a test input with the label of the nearest training point so 1-NN is parametric-free. In Centroid layer, the training points are replaced with trainable class centroids in the latent space. As a result, we are comparing the distances between the latent feature vectors encoded by the bottom network with class centroids and assign the label to the input with the label of these centroids.

Formally, recall that in Chapter 2 we use $f : \mathbb{R}^d \to \mathbb{R}^m$ to denote a neural network that takes an input $x \in \mathbb{R}^d$ and outputs a logit vector for m classes. We use the uppercase $F : \mathbb{R}^d \to [m]$ to denote the integer prediction of the network such that $F(x) = \arg\max_j f_j(x)$. To separate the Centroid layer with the bottom feature encoder, similar to Chapter 4, let $\theta = \{\theta_c, \theta_b\}$ where θ_c denotes the weights, i.e. vector

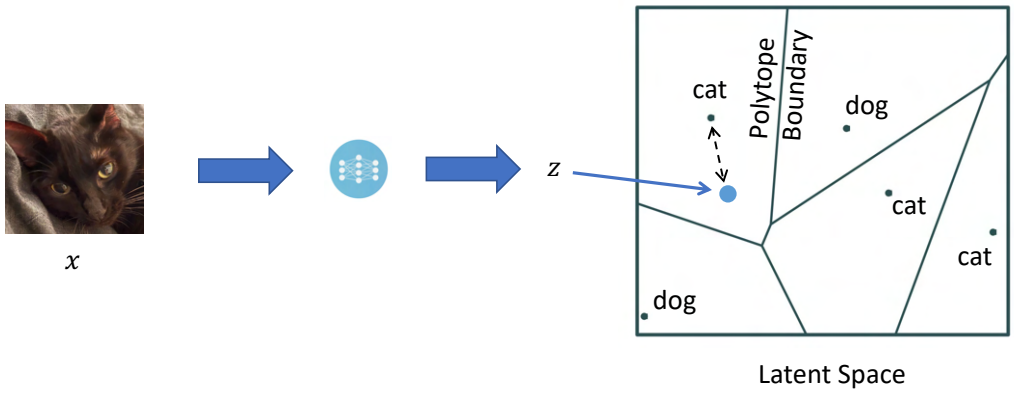


Figure 6.2: Visualization of the Voronoi diagram in the latent space of Centroid Nets with five 2-dimensional centroids.

representation of centroids, of the top Centroid layer, cnt, and θ_b denotes the weights of f until cnt. Suppose there are C centroids and the number of dimensions for each centroid is c, i.e. $\forall j \in [C], \{\theta_c\}_j \in \mathbb{R}^c$. By setting C to be fully dividable by the number of classes m, i.e. $k = C/m$ is an integer, we assign k centroids with the same class label. We will later use $\{\theta_c\}_j^{\text{label}}$ to denote the label of the j-th centroid. Finally, the forward pass of f with parameter $\theta = \{\theta_c, \theta_b\}$ is as follows,

$$f(x;\theta) = \mathsf{cnt}(z;\theta_c), \quad z = f(x;\theta_b).$$

Here $z = f(x;\theta_b)$ is the output of the penultimate layer, i.e. the latent representation of x, and $z \in \mathbb{R}^c$ so we can compute the distance between z and centroids θ_c in following cnt layer. The forward pass of cnt is inspired by a 1-NN model. The only difference is that cnt needs to output a continuous vector so we can use it as the logit of x. For this purpose, cnt will output the negative distance to each nearest class centroid as the logit. Formally, we define

$$\{\mathsf{cnt}(z;\theta_c)\}_j = \max_{\forall i, \{\theta_c\}_i^{\text{label}} = j} -\|\{\theta_c\}_i - z\|_2. \tag{6.1}$$

Geometry of the Latent Space. With cnt layer, the latent space of Centroid Nets can be partitioned into C (i.e. the number of centroids) polytopes. For any points in the same polytope, their nearest neighboring centroid is the same as illustrated in Figure 6.2. This is known as the equivalence between an 1-NN classifier and a Voronoi diagram (Senechal, 1993), a way to partition a space into sub-spaces close to each of a given set of points. Centroids are thus the Voronoi vertices in the latent space of the underlying Centroid Net. Because we are using ℓ_2 distance to measure the similarity in Equation 6.1, the resulting sub-spaces are all convex, i.e. polytopes. The boundary between a neighboring pair of polytopes is a also decision boundary

if the neighboring centroids belong to different classes.

6.3.2 Centroid Net That Opts to Reject

Distance-based classification in cnt provides a way to abstain from prediction, which we refer to as a *rejection option*, if the latent representation of the input is (1) too far away from all centroids; and (2) is pretty close to at least two centroids. Condition (1) can be seen as the case when the model is given an input that is not similar to any of known classes. Similar to humans, the model is expected to answer something equivalent to "I don't know" instead of making a prediction. Rejecting points that fall into condition (1) is therefore a desired behavior for well-aligned models. On the other hand, condition (2) corresponds to the case when the model is given an input that is similar to more than one classes at the same time. This could an image of multiple objects, e.g. dogs and cats, or an adversarial image. Rejecting images with confusing labels or adversarial features is also desired for well-aligned classifiers that are only allowed to give a single label to the input.

We now propose a rejection-based Centroid layer, i.e. η-Centroid Layer (Definition 22), to unify conditions (1) and (2) as a single rule.

Definition 22 (η-Centroid Layer) *For a standard Centroid layer, $cnt(z;\theta_c)$, defined in Equation 6.1 and let $y = \arg\max_i \{cnt(z;\theta_c)\}_i$. The output of a Centroid layer with a $\perp$ logit, $cnt^\eta(z;\theta_c)$, is equal to*

$$\forall j \in [m] \cup \{\perp\}, \{cnt^\eta(z;\theta_c)\}_j = \begin{cases} \{cnt(z;\theta_c)\}_j & \text{if } j \neq \perp \\ \frac{1-\eta}{1+\eta} \cdot \{cnt(z;\theta_c)\}_{y_{sec}} & \text{otherwise} \end{cases}, \tag{6.2}$$

$$\text{and } y_{sec} = \arg\max_{i \neq y} \{cnt(z;\theta_c)\}_i, \tag{6.3}$$

where $\eta \in [0,1]$ is a hyper-parameter for the rejection threshold.

Put it simple, η-Centroid Layer has an additional logit output of a class, which we term as $\perp$, that indicates if we should reject the input. If logit score of the $\perp$ surpasses the top logit score of a valid class, i.e. any $j \in [m]$, the Centroid Net would output $\perp$ as the prediction, which we will interpret as a rejection or "I don't know". The logit score of $\perp$ is based on the second highest logit score among the valid classes and a hyper-parameter $\eta \in [0,1]$. By the design of $\text{cnt}^\eta(z;\theta_c)$, we can prove Theorem 7.

Theorem 7 (Rejection Criteria for η-Centroid Layer) *Let $F^\eta : \mathbb{R}^d \to \mathbb{R}^m$ denote the interger prediction of a Centroid Net with a η-Centroid Layer, $cnt^\eta(z;\theta_c)$, on the top. $z = f(x;\theta_b)$ is the penultimate output of F^η and θ_c*

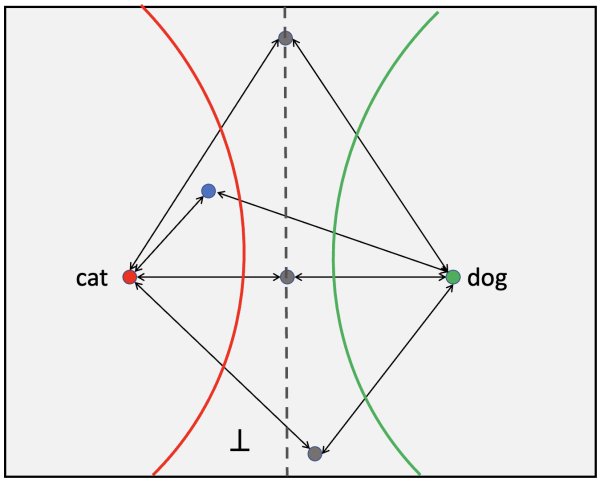


Figure 6.3: An illustration of the partitions of the latent space of a Centroid model with rejection cnt^η. The output labels of latent representations in gray are $\perp$, i.e. rejection, because of insufficient differences between distances to the nearest and the second nearest centroid. Curves in red and green are the decision boundaries for cnt^η, while the broken line is the decision boundary for a corresponding cnt layer, i.e. without rejection.

are centroids.

$$\forall x \in \mathbb{R}^d, F^\eta(x) \neq \perp \implies \|z - \{\theta_c\}_2\|_2 - \|z - \{\theta_c\}_1\|_2 \geq \eta(\|z - \{\theta_c\}_1\|_2 + \|z - \{\theta_c\}_2\|_2),$$
$$\textit{and here } z = f(x; \theta_b).$$

We use $\{\theta_c\}_1$, $\{\theta_c\}_2$ to denote the nearest centroid to z and the second-nearest centroid to z that has a different label from $\{\theta_c\}_1$, respectively.

That is, if a Centroid Net outputs a label other than $\perp$, then the nearest centroid must be close enough to the latent representation z of the input. Our design of the condition of "closeness" includes a comparison between the ratio of the difference of distances over the sum of distances with a threshold η. There are two situations will cause the difference of distances to be less than η times the sum of distances as illustrated in Figure 6.3. That is, when (1) z is far away from the top-2 closest centroid $\{\theta_c\}_1$ and $\{\theta_c\}_2$; or (2) z is close to $\{\theta_c\}_1$ and $\{\theta_c\}_2$ and the difference of distance is not distinguishable w.r.t η.

It is noticeable that cnt^η has a closed boundary between rejection and valid class, which results in a closed decision boundary in the input space. When $\eta = 0$, there is no rejection and cnt^η reduces back to a standard Centroid layer without rejection and $\eta = 1$ makes the model reject everything. Interesting, the closest distance between points on the rejection boundaries corresponding to different centroids, i.e. the red and blue ones in Figure 6.3, are the ones on the line that connects two centroids. Formally, we can prove the following Theorem 8.

Theorem 8 *Suppose $F^\eta : \mathbb{R}^d \to [m] \cup \{\bot\}$ is a Centroid Net with rejection and $f(x;\theta_b)$ is the penultimate output of F. If $f(\cdot;\theta_b)$ is K-globally Lipschitz continuous,*

$$\forall x' \in \mathbb{R}^d, \|x' - x\|_2 \leq (1+\eta)\frac{d_{min}}{K} \implies F^\eta(x') = F^\eta(x) \text{ or } F^\eta(x') = \bot \text{ or } F^\eta(x) = \bot \tag{6.4}$$
$$\text{and here } d_{min} = \min_{y\in[m], j\in\mathcal{I}_y} \left[\min_{y'\neq y, i\in\mathcal{I}_{y'}} \|\{\theta_c\}_j - \{\theta_c\}_i\|_2 \right],$$

where $\mathcal{I}_y = \{k | \{\theta_c\}_k^{label} = y\}$, a set of indices of centroids whose labels are y.

The RHS of Equation 6.4, i.e. $a = b$ or $a = \bot$ or $b = \bot$, is also referred to as a property of *global robustness* (Leino et al., 2021b) so Theorem 8 shows that a Centroid Net F^η is preserves the property of global robustness at least in the neighborhood with a radius of $(1+\eta)d_{min}/K$. It is useful to have global robustness at x because it ensures that if F^η does not reject x, the neighbors of x will either have the same predictions as x or get rejected. The radius of the ball depends both on the nearest distances between class centroids, d_{min}, and the Lipschitz constant of the feature encoder, K. This is intuitive as centroids are distant from each other, the margins between each other get greater, where the margin in the latent space is connect to the distance in the input space by the Lipschitz constant. Interestingly, based on our design of rejection and its illustration in Figure 6.3, $(1+\eta)d_{min}/K$ is actually the smallest "width" of the band between two different classes where F^η rejects the input. As a result, for points that are not on the line between any pairs of class centroids, the actual ball that one can certify global robustness for F^η is greater and the radius is the distance between x and the hyperplane that is equal distant to the pair of centroids. This property finally ensures that the decision boundary for F^η is closed in the latent space and should not extends to the infinity in contrast with conventional classifiers.

Comparing with GloRo Nets. GloRo Net (as is discussed in Chapter 5) is probably the most similar technique to Centroid Net with a rejection option. The difference is that a GloRo Net returns $\bot$ because the local robustness can not be certified at x, while a Centroid Net returns $\bot$ because the point is not close enough to the nearest centroid in the latent space. It is interesting to do do a head-to-head comparison with GloRo Nets using Centroid Nets. We include Example 2 using TwoMoons dataset as a preliminary study in this Chapter for showing the promise of Centroid Nets.

Example 2 *We train a GloRo Net and a Centroid Net with rejection on TwoMoons dataset with the training and test sets illustrated in Figure 6.4. For the test set, we inject Gaussian noise with a probability of 0.2 to make the test distribution shifted from training. We delay the details of training and architectures to Appendix* **??**. *We plot the decision boundaries of two networks on the right of Figure 6.4, where purple and green points and regions are labeled with 0 and 1, respectively. Gray regions and yellow points are labeled with $\bot$ by two networks, respectively.*

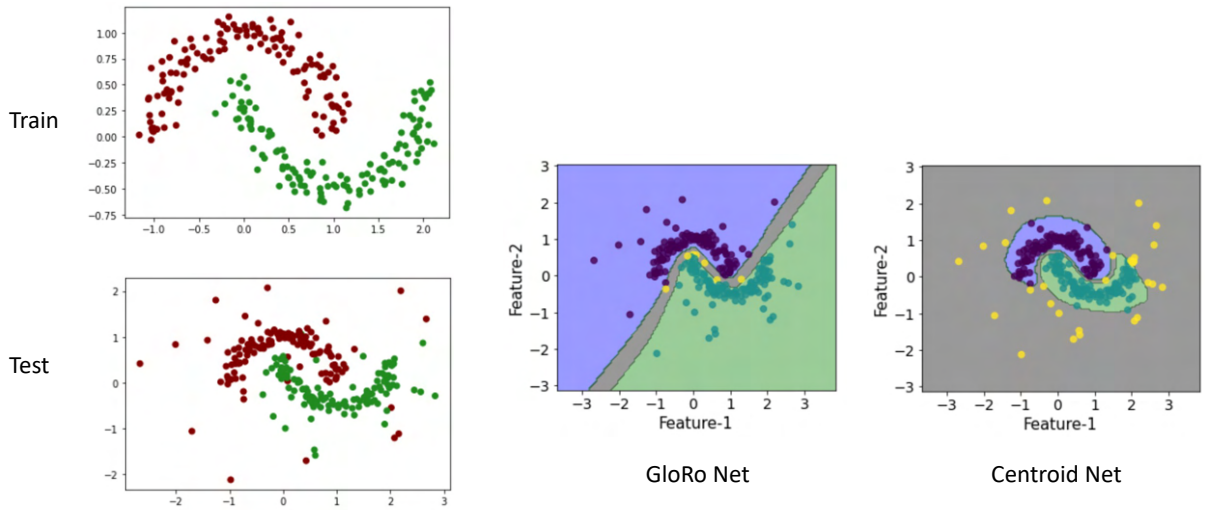


Figure 6.4: Plots of decision boundaries of a GloRo Net and a Centroid Net (with rejection) trained on TwoMoons. Yellow points are labeled as $\perp$ by the network, i.e. rejection from prediction, in both plots.

It is noticeable that in the test set shown in Figure 6.4 there are outliers for each moon, which either sit far away from the entire moons or in the middle of two halves. In the plot of GloRo Net, because the decision boundary is not closed, we see that only outliers that in the middle of two halves are rejected for the reason of the lack of local robustness. Other outlying points, even though they are perturbed and far away from moons, are classified into one of the moons robustly. On the other hand, the Centroid Net on the right in Figure 6.4 rejects these outlying noisy points which are far away from moons. In the same time, it is able to reject points in the middle of two moons as well. Care must be taken that because Centroid Net does not directly specify the radius ϵ in the input space, so the resulting minimum width of the rejection band, i.e. the narrowest place in the gray region, is smaller than that of the GloRo Net on the left. However, we argue that this can be fixed in the training when the objective function is carefully designed to entail a requirement of larger rejection band, e.g. making centroids further away. On the other hand, our plots may point out Centroid Nets are probably a better fit for tasks that have a higher tolerance for confusing better similar classes, e.g. `sports car` and `sedan`, but have a low tolerance of letting invalid input enter the system and receive a meaningful prediction.

Local Robustness of Centroid Nets. Another benefit of training a Centroid Net with rejection, F^{η}, is that its corresponding standard version, F, can be shown to preserve the property of local robustness shown in Theorem 9.

Theorem 9 *With definitions in Theorem 8, let F be the corresponding Centroid Net without rejection of F^{η}. Then, F*

is ϵ-locally robust ((Definition 11) and

$$\epsilon = (1+\eta)\frac{d_{min}}{2K}.$$

6.3.3 Self-explainability of Centroid Classification

Compared to a conventional neural network, a Centroid Net is based on the similarity between the encoded features $z = (x;\theta_b)$ and the centroids θ_c to predict the label for x. As a result, it is naturally to think θ_c as a set of *typical* representations of classes and it is interesting for the users to compare the given input with an input x_p such that $f(x_p;\theta_b)$ equals to the nearest centroid to $f(x;\theta_b)$ in the latent space. We refer to the input point, whose latent representation is the nearest centroid to the latent representation of x, as the *prototype* of x. Formally, we introduce the following Definition 23.

Definition 23 (Prototype) *Suppose θ_t is a set of centroids of a Centroid Net f and θ_b is the weights of the sub-network below the Centroid layer. The prototype x_p for a given input x is equal to*

$$x_p = \arg\max_{x'} -\|f(x';\theta_b) - \{\theta_c\}_j\|_2,$$
$$\textit{where } \{\theta_c\}_j = \arg\max_{i} -\|f(x;\theta_b) - \{\theta_c\}_i\|_2.$$

A prototype x_p of x can be used as an instance-based explanation for the prediction returned by the Centroid Net, as it gives an example point that the underlying Centroid Net believes to be *typical* for the corresponding class. On the other hand, assessing if these prototypes are perceptually meaningful helps us to understand whether the Centroid Net aligns with humans.

6.3.4 Summary

In the first part of this section, we discuss the potential of *selective classification* in closing the decision boundary for rejecting inputs that are invalid to the underlying model, adding an option of *I Don't Know* to the network. We discuss the prior works and motivate that we need a new technique that is both aware of robustness and out-of-distribution points. In the second part of this section, we introduce a new type of deep classifier, Centroid Nets, with and without an option to reject an input for a lack of confidence that the input belongs to the distribution the model is trained with. For a proof of concept, we provide a set of analytical results to describe the guarantee provided by a Centroid Net and include an example with TwoMoons dataset to illustrate the difference between a GloRo Net and of a Centroid Net. As a promising technique to realize selective classification, Centroid Nets with rejection are capable of learning closed decision boundaries and provide (global) robustness guarantee at the same time. More importantly, unlike

a conventional network, Centroid Nets are self-explainable by construction – by finding the corresponding prototype inputs of the centroids, the users can easily assess and verify whether the model has found perceptibly meaningful commonality among the training sample. Completing and improving the idea of Centroid Nets is a future work of the author for addressing the under-specification issue of deep networks to enhance alignment.

Appendix A

A Brief Review of Explainability Methods

To date, there is a large body of work delivering different types of explanations to answer different questions, raised by human users, about the model. Here we list some common questions and corresponding explanations developed in the prior works.

What is the most similar training instance that the model relies on to predict for the underlying instance? . This question is interested in the influence of the training set on the model's weights and the corresponding method is referred to as *instance-based explanation* (Das et al., 2021; Han & Tsvetkov, 2021; Jia et al., 2019; Koh & Liang, 2017; Lin et al., 2022; Pruthi et al., 2020; Schioppa et al., 2022; Yeh et al., 2018);

Where can we find a different input that can cause a different (or desired) output? The technique that answers this question is often referred to as *counterfactual explanation* (Black et al., 2022; Karimi et al., 2020; Keane & Smyth, 2020; Mahajan et al., 2019; Mothilal et al., 2020; Pawelczyk et al., 2020; Poyiadzi et al., 2020; Ustun et al., 2019; Van Looveren & Klaise, 2019; Yang et al., 2020a);

What high-level concepts can we extract from the learned representations? To answer this question, a popular approach is T-CAV (Kim et al., 2018; Yeh et al., 2020), which checks the model's internal representations for their response to well-crafted inputs that only contain specific "concepts", e.g. stripes and rectangles.

What is the most important feature to the model's inference result? The technique that answers this question is often referred to as *feature attribution*, which assigns a scalar value for each input feature for its importance towards a quantity of interest, e.g. the output logit of the top class (Binder et al., 2016; BOHNENBLUST et al., 1952; Erion et al., 2021; Fong & Vedaldi, 2017; Ghalebikesabi et al., 2021; Leino et al.,

2018; Lundberg & Lee, 2017; Pan et al., 2021; Petsiuk et al., 2018; Ribeiro et al., 2016; Selvaraju et al., 2019; Shrikumar et al., 2017; Simonyan et al., 2013; Smilkov et al., 2017; Sundararajan et al., 2017; Wang et al., 2020a, 2022).

Appendix B

Futher Details on Locality Evaluations for Non-robust Models

B.1 Implementation of Boundary Search in BIG

Our boundary search uses a pipeline of PGDs and CW Nicolae et al. (2019). PGDs is a repetition of PGD Madry et al. (2017) attack with larger ϵ until an adversarial example is found. Adversarial examples returned by each method are compared with others and closer ones are returned. If an adversarial example is not found, the pipeline will return the point from the last iteration of the first method (PGDs in our case). Hyper-parameters for each attack can be found in Table B.1. The implementation of PGDs and CW are based on Foolbox (Rauber et al., 2017, 2020). We have also tried to assemble with another attack, AutoPGD (Croce & Hein, 2020a), and find adding another attack does not improve the successful rates or decrease the distances to the boundary so we decide only rely on PGDs and CW. For details of the experiments, please refer to Appendix B.2.

B.2 An attempt of ensembling with AutoPGD

We implement AutoPGD based on the authors' public repository[1] (we only use `apgd-ce` and `apgd-dlr` losses for efficiency reasons). All computations are done using a GPU accelerator Titan RTX with a memory size of 24 GB. Comparisons on the results of the ensemble of these three approaches are shown in Fig. B.2.

Ideally, we want to have higher success rates or closer distances between the adversarial examples to the input so the attack better approximates the boundary search. However, on ResNet50, we did not find

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

PGDs	ImageNet	standard
	ϵs	[36/255., 64/255., 0.3, 0.5, 0.7, 0.9, 1.1]
	max steps	100
	step size	`adaptive`
CW	ImageNet	standard
	ϵ	1.0
	max steps	100
	step size	1e-2

Table B.1: Hyper-parameters used for adversarial attacks. `adaptive` means the actual step size is determined by $2 * \epsilon$ / max steps.

Pipeline	*Avg Distance*	*Success Rate*
(ImageNet) Standard ResNet50		
PGDs	0.549	72.1%
+ CW	0.548	72.1%
+ AutoPGD	0.548	72.1%

Table B.2: *Pipeline*: the methods used for boundary search. *Avg Distance*: the average ℓ_2 distance between the input to the boundary. *Success Rate*: the percentage when the pipeline returns an adversarial example. *Time*: per-instance time with a batch size of 64. We are using much bigger ϵs for robust models, so the success rates are higher than a standard model.

any improvement by using AutoPGD. As a result, we decide not to use this attack for the rest of models and other experiments.

B.3 Implementation of attributions

All attributions are implemented with Captum (Kokhlikyan et al., 2020). For BIG and IG, we use 20 intermediate points between the baseline and the input and the interpolation method is set to `riemann_trapezoid`.

B.4 Pre and Rec Locality Scores

In addition to Figure 3.2, we include plots of Pre (Figure B.1) and Rec (Figure B.2) locality scores with SM, IG and BIG on every models.

In Figure B.1, we find that as the accuracy improves from SqueezeNet to ViT-B16, Pre-scores are still around 0.5 in the plots of SM and IG. Furthermore, they are close to the Pre-score computed on a random model (shown by the horizontal lines on the plots). Pre-scores in the BIG plot are more different from the one computed on the random model but it is still hard to see an improvement of the precision from improved accuracy. In particular, we find two attention modesl, i.e. ViT-B16 and ViT-B32, have lower Pre-scores compared to the rest convolutional models, despite the fact they are more accurate. A lower precision indicates that more positive attribution scores are distributed outside the bounding box. That is,

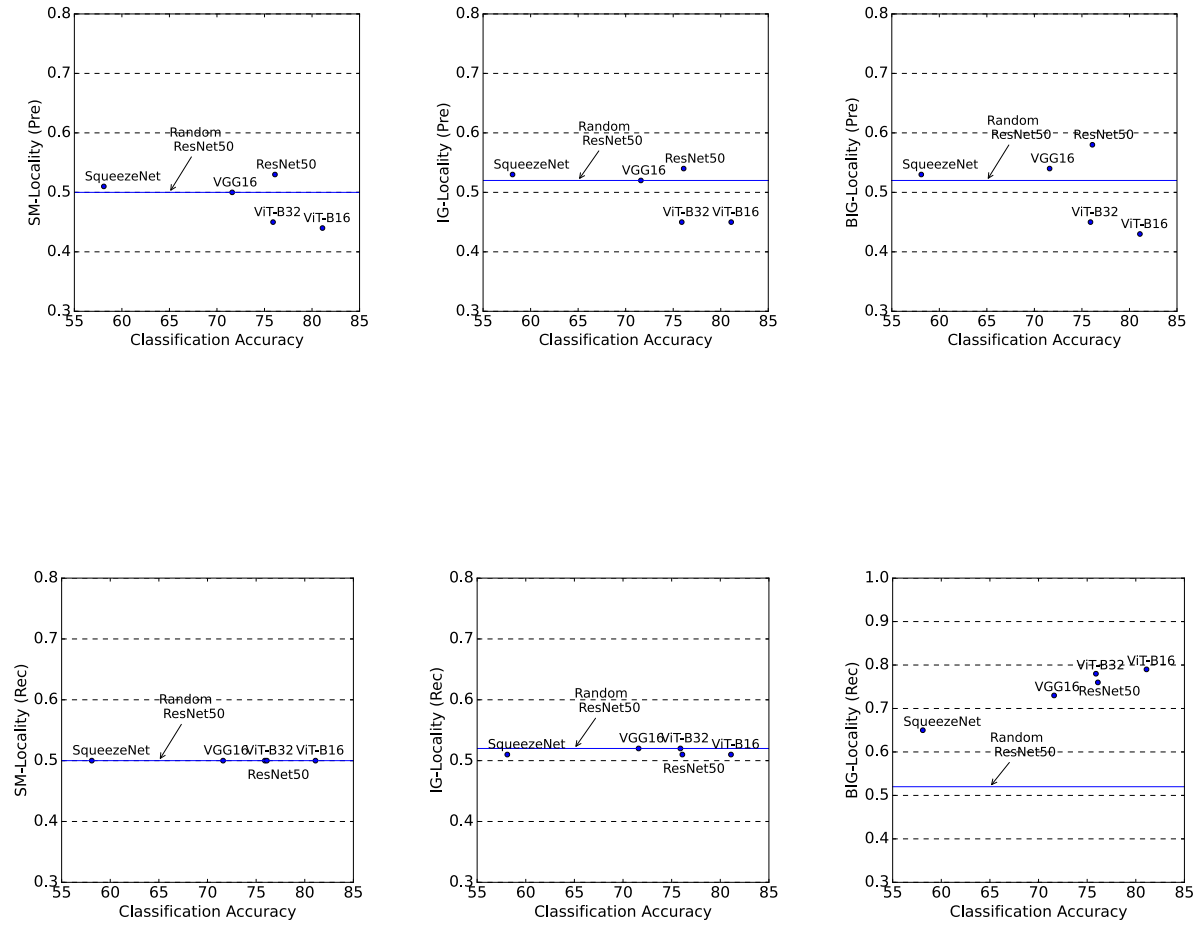


Figure B.2: Plots of Locality v.s. Top-1 Accuracy for several pre-trained models. We measure Rec-scores of locality (Definition 9) with Saliency Map (left), Integrated Gradient (middle) and Boundary-based Integrated Gradient (right). Results of averaged over 1000 images.

important features to the model are those ones that are very different from the truly important features from humans' perspective.

In Figure B.2, we find that all models have approximately 0.5 Rec-scores on locality measured with SM or IG, indicating an almost even mixture between useful and anti-useful features inside the bounding box. This observation shows that in a standard model, when a pixel is point-wise or globally important, its neighbors are very likely to have an opposite sign of importance. In the plot of BIG, we finally see some improvement of locality measured with Rec-scores as the accuracy goes up. However, because that corresponding precision is very low as shown in Figure B.1, a high recall only means a high portion of positive attributions both inside and outside the bounding box.

Appendix C

Futher Details on GloRo Nets

Chapter 5 summarizes results from our ICML-2021 paper *Globally Robust Neural Networks* (Leino et al., 2021b) and its follow-up work *Scaling in depth: Unlocking Certifiable Robustness on ImageNet* (Hu et al., 2023). Our implementation can be found at Github: `https://github.com/klasleino/gloro`. When the follow-up work (Hu et al., 2023) is public, we have updated our implementation of GloRo Nets with new hyper-parameters so the results in Chapter 5 align best with Hu et al. (2023), which we refer the implementation to if not otherwise clarified.

C.1 Implementation Details for Table 5.1

C.1.1 Training details

Dataset details The input resolution is 32 for CIFAR10/100, 64 for Tiny-ImageNet and 224 for ImageNet respectively. We apply the following data augmentation to CIFAR datasets: random cropping, RandAugment (Cubuk et al., 2020), random horizontal flipping. For Tiny-ImageNet, we find this dataset is easy to overfit and add an extra Cutout (DeVries & Taylor, 2017) augmentation. For data augmentation hyper-parameters, we use the default PyTorch setting.

Platform details Our experiments were conducted on an 8-GPU (Nvidia A100) machine with 64 CPUs (Intel Xeon Gold 6248R). Each experiment on CIFAR10/100 and Tiny-ImageNet takes one GPU and each experiment on ImageNet takes 8 GPUs. Our implementation is based on PyTorch (Paszke et al., 2019).

Training details On the first 3 datasets, all models are trained with the NAdam (Dozat, 2016) with the Lookahead optimizer wrapper (Zhang et al., 2019d) with a batch size of 256 and a learning rate of 10^{-3} for 800 epochs. We use a cosine learning rate decay (Loshchilov & Hutter, 2016) with linear warmup (Goyal

et al., 2017) in the first 20 epochs. On ImageNet, we only change the batch size to 1024 and training epochs to 400.

During training, we schedule the training ϵ to ramp up from small values and slightly overshoot the test epsilon. Let the total number of epochs be T and the test certification radius be ϵ, we use

$$\epsilon_{\text{train}}(t) = \left(\min(\frac{2t}{T}, 1) \times 1.9 + 0.1\right)\epsilon, \quad \epsilon = 36/255.$$

at epoch t. As a result, $\epsilon_{\text{train}}(t)$ begins at 0.1ϵ and increases linearly to 2ϵ before arriving halfway through the training. Later, ϵ_{train} remains 2ϵ to the end.

C.1.2 LiResNet architecture details

Model stem is used to convert the input images into feature maps. On CIFAR10/100, we use a convolution with kernel size 5, stride 2, and padding 2, followed by a MinMax activation as the stem. On Tiny ImageNet, we use a convolution with kernel size 7, stride 4, and padding 3, followed by a MinMax activation as the stem. On ImageNet, we follow the ViT-like patching (Dosovitskiy et al., 2020) and use a convolution with kernel size 14, stride 14, and padding 0, followed by a MinMax activation as the stem. Thus the output feature map size from the stem layer is 16×16 for all 4 datasets. The number of filters used in the convolution is equal to the model width W.

Model backbone is used to transform the feature maps. It is a stack of L LiResNet blocks followed by the MinMax activation, i.e., (LiResNet block $\rightarrow$ MinMax) $\times$ L. We keep the feature map resolutions and the number of channels constant in the model backbone. We find some tricks in normalization-free residual network studies (Shao et al., 2020; Zhang et al., 2019a) can improve the performance of our LiResNet as our method is also a normalization-free residual network. Specifically, we add an affine layer β that applies channel-wise learnable multipliers to each channel of the feature map (similar to the affine layer of batch normalization) and a scaler of $1/\sqrt{L}$ to the residual branch where L is the number of blocks:

$$y = x + \frac{1}{\sqrt{L}}\beta\text{Conv}(x)$$

Model neck is used to convert the feature maps into a feature vector. In our implementation, the model neck is a 2 layer network. The first layer is a convolution layer with kernel size 4, stride 4, and padding 0, followed by a MinMax activation. The number of input channels is the model width W and the number of output channels is $2W$. Then we reshape the feature map tensor into a vector. The second layer is a dense layer with output dimension d where $d = 2048$ for the three small datasets (CIFAR10/100 and Tiny-ImageNet) and $d = 4096$ for ImageNet.

Table C.1: Clean accuracy and VRA performance (%) of a ConvNet and a LiResNet on three datasets with different loss functions

loss	**TRADES**		**EMMA**	
	Clean (%)	VRA (%)	Clean (%)	VRA (%)
CIFAR-10 ($\epsilon = 36/255$, 10 classes)				
ConvNet	71.7	58.8	72.5	59.2
LiResNet	79.6	66.2	80.4	66.3
CIFAR-100 ($\epsilon = 36/255$, 100 classes)				
ConvNet	53.4	34.0	50.6	35.0
LiResNet	57.8	37.3	54.2	37.8
Tiny-ImageNet ($\epsilon = 36/255$, 200 classes)				
ConvNet	42.2	26.6	40.0	27.4
LiResNet	45.8	28.8	43.6	30.0

Model head is used to make classification predictions. We apply the last layer normalization (LLN) proposed by Singla et al. (2022) to the head.

C.1.3 Metric details

We report the clean accuracy, i.e., the accuracy without verification on non-adversarial inputs and the verified-robust accuracy (VRA), i.e., the fraction of points that are both correctly classified and certified as robust. Our results are averaged over 5 runs for CIFAR10/100 and TinyImageNet and 3 runs for ImageNet.

C.2 Details for Table 5.2

In Table 5.2a, we use an L12W256 configuration, i.e., the backbone has 12 blocks and the number of filters is 256. For ConvNet, the only difference is that the LiResNet block is replaced by a convolution of kernel 3, stride 1, and padding 1. All other settings are the same. Table C.1 is a more detailed version of Table 5.2a with clean accuracy.

In Table **??**b, we use the configuration of W256, i.e., the number of channels in the backbone is 256. The only difference between conventional ResNet and LiResNet is the block. The block for conventional ResNet is

$$y = x + \beta \text{Conv}(\text{MinMax}(\text{Conv}(x)))$$

where β is the affine layer. We find use zeros to initialize β works the best for conventional ResNet. The number of input and output channels of the two convolution layers are the same as that of the LiResNet block. Table C.2 is a more detailed version of Table 5.2b with clean accuracy.

Table C.2: Clean accuracy and VRA (%) performance on CIFAR-10/100 with different architectures (L is the number of blocks in the model backbone). We use EMMA loss for Gloro training. $\times$ stands for not converging at the end.

Dataset	L	ConvNet		ResNet		LiResNet	
		Clean(%)	VRA(%)	Clean(%)	VRA(%)	Clean(%)	VRA(%)
CIFAR-10	6	77.9	64.0	74.2	60.3	79.9	65.5
	12	72.5	59.2	74.0	60.0	80.4	66.3
	18	$\times$	$\times$	73.9	60.1	81.0	66.6
CIFAR-100	6	51.8	36.5	48.4	33.5	53.6	37.2
	12	50.6	35.0	48.1	33.5	54.2	37.8
	18	$\times$	$\times$	48.2	33.6	54.3	38.0

Table C.3: Clean accuracy and VRA (%) performance of LiResNet of different depths (L is the number of blocks in the model backbone).

L	CIFAR10		CIFAR100		Tiny-ImageNet	
	Clean(%)	VRA(%)	Clean(%)	VRA(%)	Clean(%)	VRA(%)
6	79.9	65.5	53.6	37.2	43.1	29.8
12	80.4	66.3	54.2	37.8	43.6	30.3
18	81.0	66.6	54.3	38.0	43.9	30.6
24	81.2	66.8	55.0	38.2	44.2	30.7
30	81.3	66.9	54.9	38.4	44.2	30.6
36	81.2	66.9	55.0	38.3	44.3	30.4

C.3 Optimizing for Lipschitz Lower Bounds in Table 5.3

Figure 5.3 gives empirical lower bounds on the global and average local Lipschitz constants on the models trained in our evaluation. We use optimization to obtain these lower bounds; further details are provided below.

Global Lower Bounds. We use the *margin Lipschitz constant*, K^*_{ij} which takes a different value for each pair of classes, i and j. To obtain the lower bound we optimize

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

where j_1 the predicted. Optimization is performed using Keras' default `adam` optimizer with 7,500 gradient steps. Both x_1 and x_2 are initialized to random points in the test set; we perform this optimization over 100 such initial pairs, and report the maximum value obtained over all initializations.

Local Lower Bounds. We use a variant of the *margin Lipschitz constant* (Definition **??** in Appendix **??**) analogous to the local Lipschitz constant at a point, x_0, with radius ϵ. To obtain this lower bound we

optimize

$$\max_{x_1,x_2} \max_i \left\{ \frac{\left| f_j(x_1) - f_i(x_1) - \left(f_j(x_2) - f_i(x_2)\right) \right|}{||x_1 - x_2||} \right\}$$
$$\text{subject to } ||x_1 - x_0|| \leq \epsilon,\ ||x_2 - x_0|| \leq \epsilon$$

where $j = F(x_0)$. Optimization is performed using Keras' default `adam` optimizer with 5,000 gradient steps. After each gradient step, x_1 and x_2 are projected onto the ϵ-ball centered at x_0. Both x_1 and x_2 are initialized to random points in the test set, and x_0 is a fixed random point in the test set. We perform this optimization over 100 random choices of x_0, and report the mean value.

Appendix D

Futher Details on PAC-Bayesian Bound and TrH Regularization

D.1 Theorems and Proofs

D.1.1 Proof of Theorem 4

Theorem 4 If $\mathcal{P} = \mathcal{N}(\mathbf{0}, \sigma_0^2)$, and $\mathcal{Q}$ is also a product of univariate Gaussian distributions, then the minimum of Eq. 4.5 w.r.t $\mathcal{Q}$ can be bounded by

$$\begin{aligned}
&\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D}) \\
\leq &\min_{\mathcal{Q}} \{\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^n) + \frac{1}{\beta}\mathrm{kl}(\mathcal{Q}||\mathcal{P})\} + C(\tau, \beta, m) \\
= &\min_{\theta} \{\hat{R}(\theta, D^n) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2}\mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^n))\} + C(\tau, \beta, m) + O(\sigma_0^4).
\end{aligned}$$

Proof. Before we begin our proof, we emphasize that the posterior $\mathcal{Q}$ that minimizes the test loss may not be the same posterior $\mathcal{Q}$ that minimizes the training loss. Namely, define

$$\begin{aligned}
\mathcal{Q}_{test} &= \arg\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D}) \\
\mathcal{Q}_{train} &= \arg\min_{\mathcal{Q}} \{\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^n) + \frac{1}{\beta}\mathrm{kl}(\mathcal{Q}||\mathcal{P})\},
\end{aligned}$$

then $\mathcal{Q}_{test}$ may not be equal to $\mathcal{Q}_{train}$. However, the following inequality clearly holds based on their definitions,

$$\mathbb{E}_{\theta\in\mathcal{Q}_{test}} R(\theta, \mathcal{D}) \leq \mathbb{E}_{\theta\in\mathcal{Q}_{train}} R(\theta, \mathcal{D}) \leq \mathbb{E}_{\theta\in\mathcal{Q}_{train}} \hat{R}(\theta, D^n) + \frac{1}{\beta}\mathrm{kl}(\mathcal{Q}||\mathcal{P}) + C(\tau, \beta, m).$$

Generally speaking, it is impossible to directly find $\mathcal{Q}_{test}$ without knowing the true data distribution $\mathcal{D}$. Thus, Theorem 4 attempts to derive $\mathcal{Q}_{train}$ to minimize the RHS of Eq. 4.5. When it is clear which optimal

posterior, i.e. $\mathcal{Q}_{train}$ instead of $\mathcal{Q}_{test}$, we can derive in the theorem, we omit the subscription and directly write $\mathcal{Q}$.

Proof Overview. Based on our assumptions, we write $\mathcal{Q} = \mathcal{N}(\theta, \Sigma)$ where Σ is the (diagonal) covariance matrix. The minimization of the bound w.r.t $\mathcal{Q}$ is to minimize the bound w.r.t Σ and θ. Our proof is therefore three-fold: (1) firstly, we write the expression of $\mathrm{kl}(Q||P)$ using θ and Σ; (2) secondly, we use a second-order Taylor expansion to decompose $\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^n)$ into terms as functions of Σ and θ; and (3) finally, we minimize the bound w.r.t Σ for two cases: all weights have the same or different variances. Namely, the diagonal of Σ has the same or different elements. At the end of the minimization w.r.t Σ, we arrive at the bound that requires to minimize θ to actually minimize the RHS. Our proof follows below.

Extra Notations. Throughout the proof we will use N for the total number of weights, i.e. the cardinality of θ. When indexing a particular weight, we write θ_n. Let the diagonal of Σ be $\sigma^2 = [\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2]^\top$, i.e. the variance of each weight. We use $\mathrm{Diag}(a)$ to denote the diagonal matrix where the the diagonal is the elements from vector a.

Step I: the KL term $\mathrm{kl}(\mathcal{Q}||\mathcal{P})$. The expression of the KL term $\mathrm{kl}(\mathcal{Q}||\mathcal{P})$ when $\mathcal{P} = \mathcal{N}(\mathbf{0}, \sigma_0^2 I)$ is as follows,

$$\begin{aligned}\mathrm{kl}(\mathcal{Q}||\mathcal{P}) &= \frac{1}{2}\left[\log\frac{\sigma_0^2}{\det(\Sigma)} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\mathrm{Tr}(\Sigma)}{\sigma_0^2}\right] \\ &= \frac{1}{2}\left[\sum_n \log\frac{\sigma_0^2}{\sigma_n^2} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\sum_n \sigma_n^2}{\sigma_0^2}\right] \text{(when } \Sigma \text{ is a diagonal matrix).}\end{aligned} \tag{D.1}$$

Step II: Second-order Taylor's Expansion for $\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^n)$. We expand $\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^n)$ at the point θ and use the re-parameterization trick,

$$\begin{aligned}\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^n) &= \hat{R}(\theta, D^n) + \underbrace{\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\epsilon^\top \nabla\hat{R}]}_{=0 \text{ because } \epsilon \text{ has zero mean}} + \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\epsilon^\top \nabla^2\hat{R}\epsilon] \\ &+ \underbrace{\frac{1}{6}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\sum_{i,j,k}\frac{\partial^3\hat{R}}{\partial\theta_i\theta_j\theta_k}\epsilon_i\epsilon_j\epsilon_k]}_{=0 \text{ because the mean and the skewness of a standard Gaussian are } 0} + O(\sigma^4) \\ &= \hat{R}(\theta, D^n) + \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[(\sigma\circ\epsilon)^\top \nabla^2\hat{R}(\sigma\circ\epsilon)] + O(\sigma^4)\end{aligned} \tag{D.2}$$

where $\circ$ is the element-wise product, a.k.a the Hadamard product. Notice the connection between a vector product and Hadamard product:

$$\text{for any vectors } a, b, \quad a\circ b = \mathrm{Diag}(a)b. \tag{D.3}$$

Using Eq. D.3, we simplify Eq. D.2 as follows,

$$\begin{aligned}\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta,D^n) &= \hat{R}(\theta,D^n)+\frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[(\text{Diag}(\sigma)\epsilon)^\top\nabla^2\hat{R}(\text{Diag}(\sigma)\epsilon)]+O(\sigma^4)\\ &= \hat{R}(\theta,D^n)+\frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R})+O(\sigma^4).\end{aligned} \tag{D.4}$$

Step III: Bound Minimization. Using Eq. D.1 and D.4, we re-write the bound in Theorem 3 as follows

$$\begin{aligned}E_{\theta\in\mathcal{Q}}\hat{R}(\theta,D^n) &\le \mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta,D^n)+\frac{1}{\beta}\text{kl}(\mathcal{Q}||\mathcal{P})+C(\tau,\beta,m)\\ &= \hat{R}(\theta,D^n)+\frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R})+\frac{1}{2\beta}\left[\sum_n\log\frac{\sigma_0^2}{\sigma_n^2}-N+\frac{||\theta||_2^2}{\sigma_0^2}+\frac{\sum_n\sigma_n^2}{\sigma_0^2}\right]+O(\sigma^4)+C(\tau,\beta,m).\end{aligned} \tag{D.5}$$

There are two sets of parameters in Eq. D.5 for bound minimization: Σ (i.e. the σ^2) and θ. To minimize w.r.t Σ, strictly speaking it requires minimizing all relevant terms and the higher-order terms in $O(\sigma^4)$, which is clearly infeasible. Instead, we apply the following approximation by neglecting the higher-order $O(\sigma^4)$ and solve

$$\sigma^{*2}=\arg\min_{\sigma^2}\left\{\hat{R}(\theta,D^n)+\frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R})+\frac{1}{2\beta}\left[\sum_n\log\frac{\sigma_0^2}{\sigma_n^2}-N+\frac{||\theta||_2^2}{\sigma_0^2}+\frac{\sum_n\sigma_n^2}{\sigma_0^2}\right]\right\}.$$

In the following, we discuss two cases of σ_n^2.

Case I (Spherical Gaussian): $\forall n,\sigma_n^2=\sigma_q^2$**.** In this case, the solution to the problem above is to take the derivative w.r.t. σ_q^2 and set it to zero. That is, for the optimal σ_q^{*2},

$$\text{Tr}(\hat{R}(\theta,D^n))+\frac{K}{\beta}(\frac{1}{\sigma_0^2}-\frac{1}{\sigma_q^{*2}})=0\implies\frac{\sigma_0^2}{\sigma_q^{*2}}=1+\frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta,D^n)).$$

Substituting each σ_n^2 with σ_q^{*2} in Eq. D.5 gives the solution to the following problem

$$\min_{\sigma^2}\left[\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta,D^n)+\frac{1}{\beta}\text{kl}(\mathcal{Q}||\mathcal{P})\right]=\hat{R}(\theta,D^n)+\frac{K}{2\beta}\log(1+\frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta,D^n)))+\frac{||\theta||_2^2}{2\beta\sigma_0^2}+O(\sigma_q^{*4}).$$

With the Taylor expansion for $\log(1+x)=x+O(x^2)$,

$$\frac{K}{2\beta}\log(1+\frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta,D^n)))=\frac{\sigma_0^2}{2\beta}\text{Tr}(\hat{R}(\theta,D^n))+O(\sigma_0^4)+O(\sigma_q^{*4}).$$

We can actually merge $O(\sigma_0^4)+O(\sigma_q^{*4})$ because

$$\frac{\sigma_0^2}{\sigma_q^{2*}}=1+\frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta,D^n))\implies\sigma_0^2\ge\sigma_q^{*2}$$

so $O(\sigma_0^4)+O(\sigma_q^{*4})=O(\sigma_0^4)$. Lastly, to minimize the bound, we optimize it w.r.t θ, which leads us to land on:

$$\min_\theta\{\hat{R}(\theta,D^n)+\frac{||\theta||_2^2}{2\beta\sigma_0^2}+\frac{\sigma_0^2}{2}\text{Tr}(\nabla_\theta^2\hat{R}(\theta,D^n))\}+C(\tau,\beta,m)+O(\sigma_0^4).$$

Case II (Diagonal): $\exists n_1, n_2, \sigma_{n_1}^2 \neq \sigma_{n_2}^2$. In this case, we take the derivative w.r.t each σ_n^2 and set it to zero. That is, for each optimal σ_n^{2*},

$$\frac{\partial \hat{R}^2}{\partial \theta_n^2} + \frac{1}{\beta}(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_n^{2*}}) = 0 \implies \frac{\sigma_0^2}{\sigma_n^{2*}} = 1 + \sigma_0^2 \beta \frac{\partial^2 \hat{R}}{\partial \theta_n^2}.$$

Substituting σ_n^2 with σ_n^{2*} in Eq. D.5 gives the solution to the following problem

$$\min_{\sigma_n^2} \left[\mathbb{E}_{\theta \in \mathcal{Q}} \hat{R}(\theta, D^n) + \frac{1}{\beta} \mathrm{kl}(\mathcal{Q}||\mathcal{P}) \right] = \hat{R}(\theta, D^n) + \frac{1}{2\beta} \sum_n \log(1 + \sigma_0^2 \beta \frac{\partial \hat{R}^2}{\partial \theta_n^2}) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + O(\sigma^{*4}).$$

Similar to Case I, we take Taylor's Expansion of $\log(1+x)$ so

$$\frac{1}{2\beta} \sum_n \log(1 + \sigma_0^2 \beta \frac{\partial \hat{R}^2}{\partial \theta_n^2}) = \frac{\sigma_0^2}{2} \mathrm{Tr}(\hat{R}(\theta, D^n)) + O(\sigma_0^4).$$

Lastly, using $O(\sigma_0^4) + O(\sigma^{*4}) = O(\sigma_0^4)$, we land on the same objective as derived in Case I:

$$\min_\theta \{ \hat{R}(\theta, D^n) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2} \mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^n)) \} + C(\tau, \beta, m) + O(\sigma_0^4).$$

D.1.2 Proof of Theorem 5

Theorem 5 Suppose that f is a feed-forward network with ReLU activation. For the i-th layer, let $W^{(i)}$ be its weight and $\mathcal{I}_x^{(i)} \in \mathbb{R}^{D^{(i)}}$ be its input evaluated at x. Thus, $\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)})$, i.e. TrH evaluated at x using a CE loss w.r.t $W^{(i-1)}$, is as follows

$$\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)}) = ||\{\mathcal{I}_x^{(i-1)}\}||_2^2 \sum_{k,d \in P^{(i)}} \{\mathcal{H}^{(i)}\}_{k,d}$$

$$\text{where } \{\mathcal{H}^{(i)}\}_{k,d_i} = \left[\frac{\partial \{f(x;\theta)\}_k}{\partial \{\mathcal{I}_x^{(i)}\}_{d_i}} \right]^2 \cdot h_k(x, \theta)$$

$$P^{(i)} = \{d_i | \{\mathcal{I}_x^{(i)}\}_{d_i} > 0\}$$

and the following inequality holds for any $\mathcal{H}^{(i)}, \mathcal{H}^{(i+1)}$:

$$\max_{k,d_i} \{\mathcal{H}^{(i)}\}_{k,d_i} \leq \max_{k,d_{i+1}} \mathcal{H}^{(i+1)} \cdot ||W^{(i)}||_1^2.$$

Proof. We use d_{i-1}, d_i and d_{i+1} to index the element of $\mathcal{I}_x^{(i-1)}$, $\mathcal{I}_x^{(i)}$ and $\mathcal{I}_x^{(i+1)}$. For simplicity, we write softmax$(\cdot)$ as $s(\cdot)$. By the definition of trace, we need to compute the second-order derivative of the CE loss w.r.t to each entry in $W^{(i)}$. That is, we write

$$\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)}) = \sum_{d_{i-1}, d_i} \frac{\partial^2}{\partial W_{d_{i-1}, d_i}^{(i-1)2}} (CE((x, y), \theta)). \tag{D.6}$$

First, we derive the first-order derivative. The following steps are based on the chain rule:

$$\begin{aligned}\frac{\partial}{\partial W^{(i-1)}_{d_{i-1},d_i}}(CE((x,y),\theta)) &= \sum_k \frac{\partial}{\partial\{f(x;\theta)\}_k}(CE((x,y),\theta)) \cdot \frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \\ &= \sum_k (\{s(f(x;\theta)\}_k - y_k) \cdot \frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}}.\end{aligned}$$

The second-order derivative is therefore equal to

$$\begin{aligned}\frac{\partial^2}{\partial W^{(i-1)2}_{d_{i-1},d_i}}(CE((x,y),\theta)) &= \sum_k \Big(\frac{\partial}{\partial W^{(i-1)}_{d_{i-1},d_i}} [\{s(f(x;\theta)\}_k - y_k)] \cdot \frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} + h_k(x,\theta) \cdot \frac{\partial}{\partial W^{(i-1)}_{d_{i-1},d_i}} \left[\frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right] \Big) \\ &= \sum_k \Big(h_k(x,\theta) \cdot \left[\frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right]^2 + h_k(x,\theta) \cdot \frac{\partial}{\partial W^{(i-1)}_{d_{i-1},d_i}} \left[\frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right] \Big).\end{aligned}$$

Notice that the first-order derivative of ReLU is either 1 or 0, so we have plenty of identity functions in $\frac{\partial\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}}$. For example, for any layer $j > i-1$, we can decompose this term as

$$\begin{aligned}\frac{\partial\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} &= \sum_{d_{j+1}} \frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}}} \cdot \sum_{d_j} \frac{\partial\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}}}{\partial\{\mathcal{I}_x^{(j)}\}_{d_j}} \cdot \frac{\partial\{\mathcal{I}_x^{(j)}\}_{d_j}}{\partial W^{(i-1)}_{d_{i-1},d_i}} \\ &= \sum_{d_{j+1}} \frac{\{\partial f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}}} \cdot \sum_{d_j} \mathbb{I}[\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}} > 0] \cdot W^{(j)}_{d_j,d_{j+1}} \cdot \frac{\partial\{\mathcal{I}_x^{(j)}\}_{d_j}}{\partial W^{(i-1)}_{d_{i-1},d_i}}.\end{aligned}$$

Taking derivatives of $\mathbb{I}[\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}} > 0]$ w.r.t $W^{(i-1)}_{d_{i-1},d_i}$ results in nothing but 0, so we conclude that

$$\frac{\partial}{\partial W^{(i-1)}_{d_{i-1},d_i}} \left[\frac{\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right] = 0. \tag{D.7}$$

Eq. D.7 simplifies the expression of the second-order derivative w.r.t $W^{(i-1)}_{d_{i-1},d_i}$ so we have the following clean expression:

$$\begin{aligned}\frac{\partial^2}{\partial W^{(i-1)2}_{d_{i-1},d_i}}(\mathrm{CE}((x,y),\theta)) &= \sum_k \left[\frac{\partial\{f(x;\theta)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right]^2 \cdot h_k(x,\theta) \\ &= \sum_k \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}} \cdot \frac{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}{\partial W^{(i-1)}_{d_{i-1},d_i}} \right]^2 \cdot h_k(x,\theta) \\ &= \sum_k \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}} \cdot \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{I}_x^{(i-1)}\}_{d_{i-1}} \right]^2 \cdot h_k(x,\theta) \\ &= \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{I}_x^{(i-1)}\}^2_{d_{i-1}} \cdot \sum_k \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}} \right]^2 \cdot h_k(x,\theta).\end{aligned} \tag{D.8}$$

By defining $\{\mathcal{H}^{(i)}\}_{k,d_i} = \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]^2 \cdot h_k(x,\theta)$, we further simplify Eq. D.8 as follows

$$\frac{\partial^2}{\partial W_{d_{i-1},d_i}^{(i-1)2}}(\text{CE}((x,y),\theta)) = \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{I}_x^{(i-1)}\}_{d_{i-1}}^2 \cdot \sum_k \{\mathcal{H}^{(i)}\}_{k,d_i}. \tag{D.9}$$

We plug Eq. D.9 back to Eq. D.6 and arrive

$$\begin{aligned}
\text{TrH}_x^{\text{CE}}(W^{(i-1)}) &= \sum_{d_{i-1},d_i} \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{I}_x^{(i)}\}_{d_{i-1}}^2 \cdot \sum_k \{\mathcal{H}^{(i)}\}_{k,d_i} \\
&= \Big(\sum_{d_{i-1}} \{\mathcal{I}_x^{(i-1)}\}_{d_{i-1}}^2\Big) \cdot \sum_{k,d_i} \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{H}^{(i)}\}_{k,d_i} \\
&= ||\{\mathcal{I}_x^{(i-1)}\}||_2^2 \cdot \sum_{k,d_i} \mathbb{I}[\{\mathcal{I}_x^{(i)}\}_{d_i} > 0] \cdot \{\mathcal{H}^{(i)}\}_{k,d_i}.
\end{aligned}$$

By defining $P^{(i)}$ as a set of indices of positive neurons in $\mathcal{I}_x^{(i)}$, i.e, $\forall d_2 \in P^{(i)}, \{\mathcal{I}_x^{(i)}\}_{d_2} > 0$, we simplify the equation above as follows

$$\text{TrH}_x^{\text{CE}}(W^{(i-1)}) = ||\{\mathcal{I}_x^{(i)}\}||_2^2 \cdot \sum_{k,d_i \in P^{(i)}} \{\mathcal{H}^{(i)}\}_{k,d_i}.$$

Furthermore, by the definition of $\mathcal{H}^{(i)}$, we notice that

$$\begin{aligned}
\{\mathcal{H}^{(i)}\}_{k,d_i} &= \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]^2 \cdot h_k(x,\theta) \\
&= \left\{\sum_{d_{i+1}} \left[\frac{\partial\{f(x;\theta)\}_k}{\partial\{\mathcal{I}_x^{(i+1)}\}_{d_{i+1}}} \cdot \frac{\partial\{\mathcal{I}_x^{(i+1)}\}_{d_{i+1}}}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]\right\}^2 \cdot h_k(x,\theta) \qquad \text{(D.10)} \\
&= \left\{\sum_{d_{i+1}} \left[\Big(\frac{\{\mathcal{H}^{(i+1)}\}_{k,d_{i+1}}}{h_k(x,\theta)}\Big)^{\frac{1}{2}} \cdot \frac{\partial\{\mathcal{I}_x^{(i+1)}\}_{d_{i+1}}}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]\right\}^2 \cdot h_k(x,\theta). \qquad \text{(D.11)}
\end{aligned}$$

Notice that the transition from Eq. D.10 to Eq. D.11 is because $\forall k, h_k(x,\theta) > 0$. Thus, we find

$$\{\mathcal{H}^{(i)}\}_{k,d_i} = \left\{\sum_{d_{i+1}} \left[\{\mathcal{H}^{(i+1)}\}_{k,d_{i+1}}^{\frac{1}{2}} \cdot \frac{\partial\{\mathcal{I}_x^{(i+1)}\}_{d_{i+1}}}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]\right\}^2. \tag{D.12}$$

For each layer i, we denote $\mathcal{H}_{max}^{(i)} = \max_{k,d_i}\{\mathcal{H}^{(i)}\}_{k,d_i}$. Since $\forall i, d_i, \{\mathcal{H}^{(i)}\}_{k,d_i} > 0$, we can take the square root for $\mathcal{H}_{max}^{(i)}$ such that

$$\forall k, d_i, (\mathcal{H}_{max}^{(i)})^{\frac{1}{2}} \geq (\{\mathcal{H}^{(i)}\}_{k,d_i})^{\frac{1}{2}}.$$

Thus, we can bound $\{\mathcal{H}^{(i)}\}_{k,d_i}$ as follows:

$$\{\mathcal{H}^{(i)}\}_{k,d_i} \leq \left\{ \sum_{d_{i+1}} \left[(\mathcal{H}^{(i+1)}_{max})^{\frac{1}{2}} \cdot \frac{\partial \{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial \{\mathcal{I}^{(i)}_x\}_{d_i}} \right] \right\}^2$$
$$= \mathcal{H}^{(i)}_{max} \left[\sum_{d_{i+1}} \frac{\partial \{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial \{\mathcal{I}^{(i)}_x\}_{d_i}} \right]^2 .$$

The squared term can be further bounded by using the chain rule, namely,

$$\{\mathcal{H}^{(i)}\}_{k,d_i} \leq \mathcal{H}^{(i+1)}_{max} \left[\sum_{d_{i+1}} \mathbb{I}[\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}} > 0] \cdot W^{(i)}_{d_i,d_{i+1}} \right]^2$$
$$\leq \mathcal{H}^{(i+1)}_{max} \left[\sum_{d_{i+1}} |W^{(i)}_{d_i,d_{i+1}}| \right]^2 .$$

Recall that our goal is to bound $\mathcal{H}^{(i)}_{max}$. By definition, we take the max on both sides over d_i and k (although the class dimension is already gone). The direction of the inequality still holds because quantities on both sides are all non-negative.

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max} \max_{d_i} \left[\sum_{d_{i+1}} |W^{(i)}_{d_i,d_{i+1}}| \right]^2 .$$

The order of max and square can be exchanged because $|W^{(i)}_{d_i,d_{i+1}}|$ is non-negative. Thus,

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max} \left[\max_{d_i} \sum_{d_{i+1}} |W^{(i)}_{d_i,d_{i+1}}| \right]^2 .$$

The last term inside the square is the definition of the ℓ_1 operator norm so we directly write

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max} \cdot ||W^{(i)}||^2_1.$$

D.1.3 Derivations of Propositions

Proposition 2 Given a training dataset D^n and the adversarial input example x' for each example x, the top-layer TrH of the AT loss (Definition 14) is equal to

$$\mathrm{Tr}(\nabla^2_{\theta_t} \hat{R}_{\mathtt{AT}}(\theta, D^n)) = \frac{1}{n} \sum_{(x,y)\in D^n} \mathrm{TrH}_{\mathtt{AT}}(x';\theta),$$

$$\text{where} \quad \mathrm{TrH}_{\mathtt{AT}}(x';\theta) = ||f(x';\theta_b)||^2_2 \cdot \mathbf{1}^\top h(x',\theta).$$

Proof. For simplicity, we write softmax$(\cdot)$ as $s(\cdot)$. We firstly write the expression of $\hat{R}_{\mathtt{AT}}(\theta, D^n)$ based on Definition 14,

$$\hat{R}_{\mathtt{AT}}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} \max_{||\delta||_p \leq \epsilon} \mathrm{CE}((x+\epsilon, y), \theta).$$

Let x' be an adversarial example, namely

$$x' = \arg \max_{||\delta||_p \leq \epsilon} [\text{CE}((x+\delta, y), \theta)] + x.$$

AT loss can be simplified as

$$\hat{R}_{\texttt{AT}}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} \text{CE}((x', y), \theta) = \frac{1}{n} \sum_{(x,y)\in D^n} \sum_k -y_k \log s(f'_k),$$

where $f'_k = \{\theta_t^\top f(x'; \theta_b)\}_k$ is the logit score of class k at the adversarial example x' and $s(\cdot)$ is the softmax function. Let's consider the loss at each adversarial point

$$R = \sum_k -y_k \log s(f'_k).$$

We want to compute the derivatives of R with respect to the weight at the top layer $\{\theta_t\}_{jk}$. For the ease of the notation, let's define $w = \theta_t$ so $w_{jk} = \{\theta_t\}_{jk}$ and $\nabla_{w_{jk}} R$ is equal to

$$\nabla_{w_{jk}} R = \sum_k -y_k \nabla_{w_{jk}} [\log s(f'_k)] = \{f(x'; \theta_b)\}_j (s(f'_k) - y_k). \tag{D.13}$$

The transition to Eq. D.13 uses the standard form of the gradient of Cross Entropy loss with the softmax activation. To compute the trace of a Hessian, we can skip the cross terms in Hessian (e.g. $\nabla_{w_{j_1 k}, w_{j_2 k}} R$) because they are not on the diagonal of the matrix. Thus, we are only interested in $\nabla^2_{w_{jk}} R$, which are

$$\begin{aligned}
\nabla^2_{w_{jk}} R &= \nabla_{w_{jk}} \left[\nabla_{w_{jk}} R\right] = \nabla_{w_{jk}} \left[x'_j (s(f'_k) - y_k)\right] \\
&= \{f(x'; \theta_b)\}_j \cdot \{f(x'; \theta_b)\}_j \cdot s(f'_k)(1 - s(f'_k)) \\
&= \{f(x'; \theta_b)\}_j^2 (s(f'_k) - s^2(f'_k)).
\end{aligned}$$

Eventually, to compute the trace of Hessian matrix we sum all diagonal terms so that

$$\text{Tr}(\nabla^2_{w_{jk}} R) = \sum_{j,k} \nabla^2_{w_{jk}} R = \sum_{j,k} \{f(x'; \theta_b)\}_j^2 (s(f'_k) - s^2(f'_k)) = ||f(x'; \theta_b)||_2^2 \cdot \mathbf{1}^\top h(x', \theta)$$

and we denote $\text{Tr}(\nabla^2_{w_{jk}} R)$ as $\text{TrH}_{\texttt{AT}}(x'; \theta)$ to complete the proof.

Proposition 3 Under the same assumption as in Proposition 2, the top-layer TrH of the TRADES loss (Def' 15) is equal to

$$\text{Tr}(\nabla^2_{\theta_t} \hat{R}_{\texttt{T}}(\theta, D^n)) = \frac{1}{n} \sum_{(x,y)\in D^n} \text{TrH}_{\texttt{T}}(x, x'; \lambda_t, \theta),$$

$$\text{where, } \text{TrH}_{\texttt{T}}(x, x'; \lambda_t, \theta) = ||f(x; \theta_b)||_2^2 \cdot \mathbf{1}^\top h(x, \theta) + \lambda_t ||f(x'; \theta_b)||_2^2 \cdot \mathbf{1}^\top h(x', \theta).$$

Before we start our proof of this proposition, we introduce the following useful lemmas.

Lemma 1 *Let $s(g)$ be the softmax output of the logit output g computed at input x, then*

$$\frac{\partial s(g)}{\partial g} = \Phi = diag[s(g)] - s(g) \cdot \{s(g)\}^\top \tag{D.14}$$

where $diag(v)$ returns an identity matrix with its diagonal replaced by a vector v.

Proof.

$$\frac{\partial s(g_i)}{\partial f_k} = s(g_i)(\mathbb{I}\,[i=k] - s(f_k)), \Phi_{ik} = s(g_i) \cdot \mathbb{I}\,[i=k] - s(g_i)s(f_k) \implies \frac{\partial s(g)}{\partial g} = \Phi.$$

Lemma 2 *Let $\log s(g(x))$ be the log softmax output of the logit output g computed at input x, then*

$$\frac{\partial \log s(g(x))}{\partial g(x)} = \Psi = I - \mathbf{1} \cdot \{s(g(x))\}^\top$$

where I is the identity matrix.

Proof.

$$\frac{\partial \log s(g_i)}{\partial f_k} = \mathbb{I}\,[i=k] - s(f_k), \Psi_{ik} = \mathbb{I}\,[i=k] - s(f_k) \implies \frac{\partial s(g)}{\partial g} = \Psi.$$

Now we are ready to present our proof of the Proposition 3.

Proof. We write the expression of $\hat{R}_{\mathrm{T}}(\theta, D^n)$ based on as Definition 15.

$$\hat{R}_{\mathrm{T}}(\theta, D^n) = \frac{1}{n} \sum_{(x,y)\in D^n} \Big[\mathrm{CE}((x,y),\theta) + \lambda_t \cdot \max_{||\delta||_p \le \epsilon} \mathrm{KL}((x, x+\epsilon),\theta)\Big],$$

where $\mathrm{KL}((x,x+\epsilon),\theta) = \mathrm{kl}(s(f(x;\theta))||s(\theta(x+\epsilon))$. Thus, we can compute the trace of Hessian on the top layer for the CE loss and the KL loss, respectively. Namely, we derive $\nabla^2_{\theta_t}[\mathrm{CE}((x,y),\theta)]$ and $\nabla^2[\max_{||\delta||_p\le\epsilon} \mathrm{KL}((x,x+\epsilon),\theta)]$.

First, we see that the expression of $\nabla^2_{\theta_t}[\mathrm{CE}((x,y),\theta)]$ is similar to the result in Proposition 2 by replacing the adversarial input with the clean input. With this similarity, we directly write

$$\mathrm{Tr}\{\nabla^2_{\theta_t}[\mathrm{CE}((x,y),\theta)]\} = ||f(x;\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x,\theta). \tag{D.15}$$

Second, by denoting

$$x' = \arg\max_{||\delta||_p\le\epsilon} [\mathrm{KL}((x,x+\epsilon),\theta)] + x,$$

the rest of the proof now focuses on deriving $\nabla^2_{\theta_t}[\mathrm{KL}((x,x'),\theta)]$ where

$$\mathrm{KL}((x,x'),\theta) = -\sum_i s(g_i)\log(s(g_i')) + \left[\sum_i s(g_i)\log(s(g_i))\right], \quad g_i = \{\theta_t^\top f(x;\theta_b)\}_i, g_i' = \{\theta_t^\top f(x';\theta_b)\}_i.$$

For the ease of the notation, let $w = \theta_t$ so $w_{jk} = \{\theta_t\}_{jk}$ and let $K = \mathrm{KL}((x, x'), \theta)$. To find $\nabla^2_{\theta_t} K$, we first write the first-order derivative of K with respect to w,

$$\frac{\partial K}{\partial w_{jk}} = -\sum_i s(g_i) \frac{\partial}{\partial f'_k} \left[\log(s(g'_i))\right] \frac{\partial f'_k}{\partial \partial w_{jk}} + \underbrace{\sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial f_k} \frac{\partial f_k}{\partial \partial w_{jk}}}_{\text{gradient through } f_k}. \tag{D.16}$$

Next, we discuss two cases depending on whether or not we stop gradient on g (the logit output of the clean input) in Eq. D.16. In Case I where the gradient on g is stopped, the second term of Eq. D.16 will vanish. This leads to a simpler but practically more stable objective function.

Case I: Stop Gradient on g. In this case,

$$\begin{aligned}\frac{\partial K}{\partial w_{jk}} &= -\sum_i s(g_i) \frac{\partial}{\partial f'_k} \left[\log(s(g'_i))\right] \frac{\partial f'_k}{\partial \partial w_{jk}} \\ &= \{f(x'; \theta_b)\}_j (s(f'_k) - s(f_k)).\end{aligned} \tag{D.17}$$

By comparing Eq. D.17 with Eq. D.13 and treating $s(f_k)$ as constants, we can quickly write out the second-order derivative as

$$\frac{\partial^2 K}{\partial w^2_{jk}} = \{f(x'; \theta_b)\}^2_j (s(f'_k) - s^2(f'_k)).$$

Recalling $h(x', \theta) = s(g') - s^2(g')$, therefore,

$$\mathrm{Tr}\{\nabla^2_{\theta_t}[\mathrm{KL}((x, x'), \theta)]\} = \mathrm{Tr}\{\nabla^2_{\theta_t} K\} = \sum_{jk} \frac{\partial^2 K}{\partial w^2_{jk}} = ||f(x'; \theta_b)||^2_2 \cdot \mathbf{1}^\top h(x', \theta). \tag{D.18}$$

Finally, we combine Eq. D.15 and D.18 to arrive at

$$\mathrm{Tr}\Big\{\nabla^2_{\theta_t}\Big[\mathrm{CE}((x, y), \theta) + \lambda_t \cdot \max_{||\delta||_p \le \epsilon} \mathrm{KL}((x, x + \epsilon), \theta)\Big]\Big\} = ||f(x; \theta_b)||^2_2 \cdot \mathbf{1}^\top h(x, \theta) + \lambda_t ||f(x'; \theta_b)||^2_2 \cdot \mathbf{1}^\top h(x', \theta),$$

$$\text{where } x' = \arg\max_{||\delta||_p \le \epsilon} [\mathrm{KL}((x, x + \epsilon), \theta)] + x$$

By denoting $\mathrm{TrH_t}(x, x'; \lambda_t, \theta) = \mathrm{Tr}\Big\{\nabla^2_{\theta_t}\Big[\mathrm{CE}((x, y), \theta) + \lambda_t \cdot \max_{||\delta||_p \le \epsilon} \mathrm{KL}((x, x + \epsilon), \theta)\Big]\Big\}$, we complete the proof for this case and this is the statement shown in Proposition 3.

Case II: With Gradient on g. We restart our derivation from Eq. D.16 and expand the first term as follows

$$\frac{\partial K}{\partial w_{jk}} = \{f(x'; \theta_b)\}_j (s(f'_k) - s(f_k)) + \sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial f_k} \frac{\partial f_k}{\partial \partial w_{jk}}.$$

Here

$$\sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial f_k} \frac{\partial f_k}{\partial \partial w_{jk}} = -\{f(x; \theta_b)\}_j \sum_i \frac{\partial s(g_i)}{\partial f_k} \log s(g'_i) + \{f(x; \theta_b)\}_j \sum_i \left[\frac{\partial s(g_i)}{\partial f_k} \log s(g_i) + s(g_i) \frac{\partial \log s(g_i)}{\partial f_k}\right]$$

Using Lemma 1 and 2, we write

$$\frac{\partial s(g_i)}{\partial f_k} = \Phi_{ik}, \qquad \frac{\partial \log s(g_i)}{\partial f_k} = \Psi_{ik}, \quad \Phi_{ik} = s(g_i)\Psi_{ik}$$

and

$$\sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial f_k} \frac{\partial f_k}{\partial \partial w_{jk}} = -\{f(x;\theta_b)\}_j \sum_i \Phi_{ik} \log s(g_i') + \{f(x;\theta_b)\}_j \sum_i [\Phi_{ik} \log s(g_i) + \Phi_{ik}] .$$

Therefore, the first-order derivative of K with respect to w_{jk} is

$$\begin{aligned}\frac{\partial K}{\partial w_{jk}} &= \{f(x';\theta_b)\}_j (s(f_k') - s(f_k)) - \{f(x;\theta_b)\}_j \sum_i \Phi_{ik} \log s(g_i') + \{f(x;\theta_b)\}_j \sum_i [\Phi_{ik} \log s(g_i) + \Phi_{ik}] \\ &= \underbrace{\{f(x';\theta_b)\}_j s(f_k')}_{K'} \underbrace{-\{f(x';\theta_b)\}_j s(f_k)}_{K_1} \underbrace{-\{f(x;\theta_b)\}_j \sum_i \Phi_{ik} \log s(g_i')}_{K_2} + \underbrace{\{f(x;\theta_b)\}_j \sum_i [\Phi_{ik} \log s(g_i) + \Phi_{ik}]}_{K_3} .\end{aligned}$$

To calculate the seconder-order derivative of K w.r.t w_{jk}, we find the derivative of K', K_1, K_2, K_3 w.r.t w_{jk}, respectively.

(1) Derivative of K'. K' is simply the result we have already obtained in Case I (see Eq. D.17); therefore,

$$\frac{\partial K'}{\partial w_{jk}} = \{f(x';\theta_b)\}_j^2 (s(f_k') - s^2(f_k')).$$

Thus,

$$\mathrm{Tr}(\frac{\partial K'}{\partial w_{jk}}) = ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta).$$

(2) Derivative of K_1.

$$\begin{aligned}\frac{\partial K_1}{\partial w_{jk}} &= -\{f(x';\theta_b)\}_j \{f(x;\theta_b)\}_j \frac{\partial s(f_k)}{\partial f_k} \\ &= -\{f(x';\theta_b)\}_j \{f(x;\theta_b)\}_j \Phi_{kk} \\ &= -\{f(x';\theta_b)\}_j \{f(x;\theta_b)\}_j (s(f_k) - s^2(f_k)).\end{aligned}$$

Thus,

$$\begin{aligned}\mathrm{Tr}(\frac{\partial K_1}{\partial w_{jk}}) &= -\sum_{jk} \{f(x';\theta_b)\}_j \{f(x;\theta_b)\}_j (s(f_k) - s^2(f_k)) \\ &= -f(x';\theta_b)^\top f(x;\theta_b) \cdot \mathbf{1}^\top h(x,\theta)\end{aligned}$$

(3) Derivative of K_2.

$$\begin{aligned}\frac{\partial K_2}{\partial w_{jk}} &= -\{f(x;\theta_b)\}_j \sum_i \left[\frac{\partial \Phi_{ik}}{\partial f_k} \frac{\partial f_k}{\partial w_{jk}} \log s(g_i') + \Phi_{ik} \frac{\partial \log s(g_i')}{\partial f_k'} \frac{\partial f_k'}{\partial w_{jk}} \right] \\ &= -\{f(x;\theta_b)\}_j \sum_i \left[\frac{\partial \Phi_{ik}}{\partial f_k} \{f(x;\theta_b)\}_j \log s(g_i') + \Phi_{ik} \Psi_{ik}' \{f(x';\theta_b)\}_j \right]\end{aligned}$$

Notice that

$$\begin{aligned} i = k \implies \frac{\partial \Phi_{ik}}{\partial f_k} &= \frac{\partial \Phi_{kk}}{\partial f_k} = \frac{\partial}{\partial f_k}(s(f_k) - s^2(f_k)) = \Phi_{kk} - 2s(f_k)\Phi_{kk}; \\ \text{and } i \neq k \implies \frac{\partial \Phi_{ik}}{\partial f_k} &= \frac{\partial}{\partial f_k}(-s(g_i)s(f_k)) = -2s(g_i)\Phi_{kk}. \end{aligned}$$

Thus,

$$\frac{\partial \Phi_{ik}}{\partial f_k} = \Phi_{kk}(\mathbb{I}[k=i] - s(g_i)) = \Phi_{kk}\{\Psi^\top\}_{ik} = \Phi_{kk}\Psi_{ki}.$$

As a result,

$$\begin{aligned} \frac{\partial K_2}{\partial w_{jk}} &= -\{f(x;\theta_b)\}_j \sum_i \left[\Phi_{kk}\Psi_{ki}\{f(x;\theta_b)\}_j \log s(g_i') + \Phi_{ik}\Psi'_{ik}\{f(x';\theta_b)\}_j\right] \\ &= -\{f(x;\theta_b)\}_j^2 \Phi_{kk} \sum_i \Psi_{ki} \log s(g_i') - \{f(x;\theta_b)\}_j\{f(x';\theta_b)\}_j \sum_i \Phi_{ik}\Psi'_{ik} \\ &= (-\{f(x;\theta_b)\}_j^2 \Phi_{kk})(\Psi_k \cdot \log s(g')) - \{f(x;\theta_b)\}_j\{f(x';\theta_b)\}_j(\{\Phi^\top\}_k \cdot \{\Psi'^\top\}_k). \end{aligned}$$

Thus,

$$\begin{aligned} \mathrm{Tr}(\frac{\partial K_2}{\partial w_{jk}}) &= \sum_{jk}\left[(-\{f(x;\theta_b)\}_j^2 \Phi_{kk})(\Psi_k \cdot \log s(g')) - \{f(x;\theta_b)\}_j\{f(x';\theta_b)\}_j(\{\Phi^\top\}_k \cdot \{\Psi'^\top\}_k)\right] \\ &= -||f(x;\theta_b)||_2^2 \cdot \psi'^\top h(x,\theta) - f_{\theta_b}^\top(x') f(x;\theta_b) \cdot \mathbf{1}^\top \omega', \end{aligned}$$

where ψ', ω' are vectors such that $\psi'_k = \Psi_k \cdot \log s(g'), \omega'_k = \{\Phi^\top\}_k \cdot \{\Psi'^\top\}_k$.

(4) Derivative of K_3.

$$\begin{aligned} \frac{\partial K_3}{\partial w_{jk}} &= \{f(x;\theta_b)\}_j \sum_i \left[\frac{\partial \Phi_{ik}}{\partial f_k}\frac{\partial f_k}{\partial w_{jk}} \log s(g_i) + \Phi_{ik}\frac{\partial \log s(g_i)}{\partial f_k}\frac{\partial f_k}{\partial w_{jk}} + \frac{\partial \Phi_{ik}}{\partial f_k}\frac{\partial f_k}{\partial w_{jk}}\right] \\ &= \{f(x;\theta_b)\}_j^2 \sum_i \left[\Phi_{kk}\Psi_{ki}\log s(g_i) + \Phi_{ik}\Psi_{ik} + \Phi_{kk}\Psi_{ki}\right] \\ &= \{f(x;\theta_b)\}_j^2 \left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k) + \Phi_{kk}(\underbrace{\Psi_k \cdot \mathbf{1}}_{\text{row sum is 0}})\right] \\ &= \{f(x;\theta_b)\}_j^2 \left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k)\right]. \end{aligned}$$

Thus,

$$\begin{aligned} \mathrm{Tr}(\frac{\partial K_3}{\partial w_{jk}}) &= \sum_{jk}\left[\{f(x;\theta_b)\}_j^2 \left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k)\right]\right] \\ &= ||f(x;\theta_b)||_2^2 \cdot (\psi^\top h(x,\theta) + \mathbf{1}^\top \omega). \end{aligned}$$

Finally, we have

$$\begin{aligned}\mathrm{Tr}(\frac{\partial^2 K}{\partial w_{jk}^2}) &= \mathrm{Tr}\left[\frac{\partial K'}{\partial w_{jk}} + \frac{\partial K_1}{\partial w_{jk}} + \frac{\partial K_2}{\partial w_{jk}} + \frac{\partial K_3}{\partial w_{jk}}\right] \\ &= ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta) + G(x,x';\theta)\end{aligned}$$

where

$$\begin{aligned}G(x,x';\theta) = &-f(x';\theta_b)^\top f(x;\theta_b)\cdot \mathbf{1}^\top h(x,\theta) - f(x';\theta_b)^\top f(x;\theta_b)\cdot \mathbf{1}^\top h(x,\theta) - ||f(x;\theta_b)||_2^2 \cdot \psi'^\top h(x,\theta) \\ &- f_{\theta_b}^\top(x') f(x;\theta_b)\cdot \mathbf{1}^\top \omega' + ||f(x;\theta_b)||_2^2 \cdot (\psi^\top h(x,\theta) + \mathbf{1}^\top \omega).\end{aligned}$$

We arrive at

$$\begin{aligned}&\mathrm{Tr}\Big\{\nabla_{\theta_t}^2\Big[\mathrm{CE}((x,y),\theta) + \lambda_t \cdot \max_{||\delta||_p\le\epsilon} \mathrm{KL}((x,x+\epsilon),\theta)\Big]\Big\} \\ &\qquad = ||f(x;\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x,\theta) + \lambda_t(||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta) + G(x,x';\theta)),\end{aligned} \tag{D.19}$$

where

$$x' = \arg\max_{||\delta||_p\le\epsilon} [\mathrm{KL}((x,x+\epsilon),\theta)] + x$$

Finally, we complete the derivation by denoting $\mathrm{TrH}_{\mathtt{t}}(x,x';\lambda_t,\theta) = \mathrm{Tr}\Big\{\nabla_{\theta_t}^2\Big[\mathrm{CE}((x,y),\theta) + \lambda_t \cdot \max_{||\delta||_p\le\epsilon} \mathrm{KL}((x,x+\epsilon),\theta)\Big]\Big\}$.

D.1.4 TrH Regularization for Other Adversarial Losses

In this section, we apply TrH regularization to some additional robust losses other than AT or TRADES. Similarly, we use s for for simplicity.

ALP (Kannan et al., 2018). Similar to TRADES, Adversarial Logit Pairing (ALP) (Kannan et al., 2018) is another method that pushes points away from the decision boundary by regularizing the ℓ_2 difference between the clean and the adversarial softmax outputs. Formally, ALP minimizes the following loss during training,

$$\begin{aligned}\hat{R}_{\mathtt{A}}(\theta, D^n) &= \frac{1}{n}\sum_{(x,y)\in D^n} \mathrm{CE}((x',y),\theta) + \lambda_A ||s(f(x;\theta)) - s(f(x';\theta))||_2^2 \\ \text{where } x' &= x + \arg\max_{||\delta||_p\le\epsilon} \mathrm{CE}((x+\epsilon,y),\theta).\end{aligned}$$

We hereby derive TrH regularization for $\hat{R}_{\mathtt{A}}(\theta, D^n)$. Using Proposition 2, we know that

$$\mathrm{TrH}(\mathrm{CE}((x',y),\theta)) = ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta).$$

The rest of this section will focus on the ℓ_2 distance loss,

$$S = ||s(f(x;\theta)) - s(f(x';\theta))||_2^2 = \sum_i (s(g_i) - s(g_i'))^2, \tag{D.20}$$

where $g = f(x;\theta), g' = f(x';\theta)$. To compute $\mathrm{Tr}(\nabla^2_{\theta_t} S)$, we need to re-use Lemma 1 and 2 to obtain the derivatives of the softmax and log-softmax outputs, i.e.

$$\frac{\partial s(f(x;\theta))}{f(x;\theta)} = \Phi = \mathrm{Diag}(s(f(x;\theta))) - s(f(x;\theta)) \cdot s(f(x;\theta))^\top$$
$$\text{and} \quad \frac{\partial \log s(f(x;\theta))}{f(x;\theta)} = \Psi = I - \mathbf{1} \cdot s(f(x;\theta))^\top,$$

as well as

$$\frac{\partial \Phi_{ik}}{\partial \{g_\theta(x)\}_k} = \Phi_{kk}\Psi_{ki}.$$

For the ease of notation, we let $w = \theta_t$ be the weights of the top-layer. Now we are ready to write the first-order derivative of S w.r.t w_{jk} as follows

$$\begin{aligned}\frac{\partial S}{\partial w_{jk}} &= \sum_i 2(s(g_i) - s(g_i')) \left[\frac{\partial s(g_i)}{\partial w_{jk}} - \frac{\partial s(g_i')}{\partial w_{jk}}\right] \\ &= 2\sum_i (s(g_i) - s(g_i'))(\Phi_{ik}\{f(x;\theta_b)\}_j - \Phi'_{ik}\{f(x';\theta_b)\}_j).\end{aligned}$$

The second-order derivative is equal to

$$\begin{aligned}\frac{\partial^2 S}{\partial w_{jk}^2} &= 2\sum_i \left[\frac{\partial s(g_i)}{\partial w_{jk}} - \frac{\partial s(g_i')}{\partial w_{jk}}\right](\Phi_{ik}\{f(x;\theta_b)\}_j - \Phi'_{ik}\{f(x';\theta_b)\}_j) \\ &\quad + (s(g_i) - s(g_i'))\left[\frac{\partial \Phi_{ik}}{\partial w_{jk}}\{f(x;\theta_b)\}_j - \frac{\partial \Phi'_{ik}}{\partial w_{jk}}\{f(x';\theta_b)\}_j\right] \qquad \text{(D.21)} \\ &= 2\sum_i (\Phi_{ik}\{f(x;\theta_b)\}_j - \Phi'_{ik}\{f(x';\theta_b)\}_j)^2 \\ &\quad + (s(g_i) - s(g_i'))(\Phi_{kk}\Psi_{ki}\{f(x;\theta_b)\}_j^2 - \Phi'_{kk}\Psi'_{ki}\{f(x';\theta_b)\}_j^2). \qquad \text{(D.22)}\end{aligned}$$

Similar to TRADES, if one stops gradients over the clean logit g, then Eq. D.22 can be simplified as,

$$\begin{aligned}\frac{\partial^2 S}{\partial w_{jk}^2} &= 2\sum_i (\Phi'_{ik}\{f(x';\theta_b)\}_j)^2 + (s(g_i) - s(g_i'))(-\Phi'_{kk}\Psi'_{ki}\{f(x';\theta_b)\}_j^2) \\ &= 2\{f(x';\theta_b)\}_j^2 \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k)\Big).\end{aligned}$$

Therefore, the trace of Hessian is equal to,

$$\begin{aligned}\mathrm{Tr}(\nabla^2_{\theta_t} S) &= \sum_{jk} 2\{f(x';\theta_b)\}_j^2 \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k) \\ &= 2||f(x';\theta_b)||_2^2 \cdot \sum_k \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k)\Big).\end{aligned}$$

If the gradient over g is kept, then one can sum over the feature dimension j and the class dimension k in Eq. D.22. To put together, the TrH regularization term for ALP is given as follows:

$$\frac{1}{n}\sum_{(x,y)\in D^n} ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta) + \lambda_A \cdot \mathrm{Tr}(\nabla^2_{\theta_t} S).$$

MART (Wang et al., 2020b). In (Wang et al., 2020b), Wang et al. proposed MART as a robust training loss that focus more on points that are not classified correctly. Different from AT, TRADES and ALP, MART explicitly aims to increase the margin between the top prediction and the second best candidate. In what follows we provide the TrH regularization for MART. We denote the MART loss as $\hat{R}_{\mathtt{M}}$, which contains two components: a boosted Cross-Entropy (BCE) loss and a weighted KL-Divergence (WKL),

$$\hat{R}_{\mathtt{M}}(\theta, D^n) = \frac{1}{n}\sum_{(x,y)\in D^n} \mathrm{BCE}((x',y),\theta) + \lambda_m((x,x'),\theta),$$

where

$$\mathrm{BCE}((x',y),\theta) = \mathrm{CE}((x,y),\theta) + (-\log(1-\max_{\kappa\neq y} s(\{f(x';\theta)\}_\kappa)),$$
$$\text{and, } ((x,x'),\theta) = \mathrm{kl}(s(f(x;\theta))||s(f(x';\theta)))(1-s(\{f(x;\theta))\}_y).$$

Here

$$x' = x + \arg\max_{||\delta||_p\leq\epsilon} \mathrm{CE}((x,y),\theta). \tag{D.23}$$

First, we derive TrH of the BCE loss with respect to the top-layer weights θ_t,

$$\mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{BCE}) = \mathrm{Tr}(\nabla^2_{\theta_t}[\mathrm{CE}((x,y),\theta)]) + \mathrm{Tr}(\nabla^2_{\theta_t}[-\log(1-\max_{\kappa\neq y} s(\{f(x';\theta)\}_\kappa)]).$$

Using intermediate steps from the proof of Proposition 3 in Appendix D.1, we have that

$$\mathrm{Tr}(\nabla^2_{\theta_t}[\mathrm{CE}((x,y),\theta)]) = ||f(x;\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x,\theta).$$

In order to compute $\mathrm{Tr}(\nabla^2_{\theta_t}[-\log(1-\max_{\kappa\neq y} s(\{f(x';\theta)\}_\kappa)])$, we make use of Lemma 1 and 2 to obtain the derivatives of the softmax and log-softmax outputs,

$$\frac{\partial s(f(x';\theta))}{\partial f(x';\theta)} = \Phi' = \mathrm{Diag}(s(f(x';\theta))) - s(f(x';\theta))\cdot s(f(x';\theta))^\top$$
$$\text{and} \quad \frac{\partial \log s(f(x';\theta))}{\partial f(x';\theta)} = \Psi' = I - \mathbf{1}\cdot s(f(x';\theta))^\top,$$

as well as,

$$\frac{\partial \Phi'_{ik}}{\partial \{g_\theta(x')\}_k} = \Phi'_{kk}\Psi'_{ki}.$$

Let us denote $K = -\log(1 - \max_{\kappa \neq y} s(\{f(x';\theta)\}_\kappa)]$ and $w = \theta_t$ for the ease of notation. Then the first-order derivative of K w.r.t. w_{jk} is,

$$\frac{\partial K}{\partial w_{jk}} = \frac{\Phi'_{\kappa^* k}}{1 - s(\{f(x';\theta)\}_{\kappa^*})} \{f(x';\theta_b)\}_j,$$

where

$$\kappa^* = \arg\max_{\kappa \neq y} s(\{f(x';\theta)\}_\kappa).$$

The second-order derivative is,

$$\begin{aligned}
\frac{\partial^2 K}{\partial w_{jk}^2} &= \frac{\frac{\partial \Phi'_{\kappa^* k}}{\partial w_{jk}}(1 - s(\{f(x';\theta)\}_{\kappa^*})) + \Phi'_{\kappa^* k}\Phi'_{\kappa^* k}\{f(x';\theta_b)\}_j}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \{f(x';\theta_b)\}_k \\
&= \frac{\Phi'_{kk}\Psi'_{k\kappa^*}\{f(x';\theta_b)\}_k(1 - s(\{f(x';\theta)\}_{\kappa^*})) + \Phi'_{\kappa^* k}\Phi'_{\kappa^* k}\{f(x';\theta_b)\}_j}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \{f(x';\theta_b)\}_j \\
&= \{f(x';\theta_b)\}_j^2 \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{f(x';\theta)\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \right].
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathrm{Tr}\left(\frac{\partial^2 K}{\partial w_{jk}^2}\right) &= \sum_{jk} \{f(x';\theta_b)\}_j^2 \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{f(x';\theta)\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \right] \\
&= ||f(x';\theta_b)||_2^2 \sum_k \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{f(x';\theta)\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \right],
\end{aligned}$$

and

$$\mathrm{Tr}(\nabla^2_{\theta_t} \mathrm{BCE}) = ||f(x;\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x,\theta) + ||f(x';\theta_b)||_2^2 \sum_k \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{f(x';\theta)\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{f(x';\theta)\}_{\kappa^*}))^2} \right].$$

Next, we derive the trace of Hessian for WKL loss. Similarly to before, by stopping the gradient over the clean logit $f(x;\theta)$, the trace of Hessian of WKL will be the same as the KL loss used in TRADES, which has been shown in Proposition 3. Namely, in this case

$$\mathrm{Tr}(\nabla^2_{\theta_t}) = ||f(x';\theta_b)||_2^2 \cdot \mathbf{1}^\top h(x',\theta).$$

On the other hand, if the gradient on $f(x;\theta)$ is computed, there is an additional term in $\mathrm{Tr}(\nabla^2_{\theta_t})$ (similar but more complicated than $G(x,x';\theta)$ derived in the proof of Proposition 3 in Appendix D.1), which we will leave as future work. Finally, we present the TrH regularization for MART as follows,

$$\mathrm{TrH_M}(x,x';\lambda_m,\theta) = \mathrm{Tr}(\nabla^2_{\theta_t} \mathrm{BCE}) + \lambda_m \mathrm{Tr}(\nabla^2_{\theta_t}).$$

This concludes our derivations of the TrH regularization for the MART loss.

Pre-Selected Hyper-parameters (CIFAR-10/100)

Parameter	Reason To Select	Used
`img_size`	to be compatible with ϵ	32×32
`patch_size`	same as Mahmood et al. (2021)	4×4
`parameter_init`	publicly available	ImageNet21K
`batch_size`	memory	768
`warm_up iterations`	standard	500

Pre-tuning Stage

Parameter	Range	Explanation	Used
`optimizer`	{'sgd+momentum', 'adam'}	optimizer	sgd+momentum
`base_lr`	$\{0.001, 0.01, 0.1\}$	initial learning rate	0.1
`l2_reg`	$\{0, 0.0001, 0.001\}$	coefficient for L2 regularization	0
`data_aug`	{'fb', 'crop'}	data augmentation method 'fb': flip and random brightness 'crop': randomly crop and upsample	'fb'
`downsample`	{'cubic', 'nearest', 'bilinear' }	downsample method for the first kernel to fit the patch size from 16×16 to 4×4	cubic
`patch_stride`	$\{2, 4\}$	the stride to create image patches	2
`data_range`	$\{[-1, 1], [0, 1]$, 'centered'}	data range 'centered': 0-mean and 1-std	centered
`use_cutmix`	{True, False}	whether to use cutmix augmentation	False

Table D.1: Frozen Hyper-parameters for CIFAR-10 and CIFAR-100 (including DDPM images) in All Experiments.

Pre-Selected Hyper-parameters (ImageNet)

Parameter	Reason To Select	Used
`data_range`	to align with ImageNet21K checkpoint	$[-1, 1]$
`img_size`	to be compatible with ϵ	224×224
`patch_size`	standard	16×16
`parameter_init`	publicly available	ImageNet21K checkpoint
`batch_size`	memory	64
`warm_up iterations`	standard	500

Pre-tuning Stage

Parameter	Range	Explanation	Final Frozen Value
`optimizer`	{'sgd+momentum', 'adam'}	optimizer	'sgd+momentum'
`base_lr`	$\{0.001, 0.01, 0.1\}$	initial learning rate	0.01
`decay_type`	{'multistep', 'cosine'}	the function used to schedule lr decay	cosine
`l2_reg`	$[0.0001, 0.001]$	coefficient for ℓ_2 regularization	0.0001
`data_aug`	{'fb', 'crop'}	data augmentation method 'fb': flip and random brightness 'crop': randomly crop and upsample	'fb'
`use_cutmix`	{True, False}	whether to use cutmix augmentation	False

Table D.2: Frozen Hyper-parameters for ViT-B16 and ViT-L16 to reproduce results in Table. 4.1.

D.2 Hyper-parameter Pre-Tuning

Pre-Tuning. Training ViTs can sometimes be challenging due to a large amount of hyper-parameters. For the choice of the parameters that are shared across different defense methods, e.g. batch size, patch size, training iterations, and etc., we do a large grid search and choose the parameter setting that produce the best results on TRADES(base) and use it for all the methods. This step is referred to as pre-tuning and is

done per-dataset.

Pre-selected Hyper-parameters. There is a set of hyper-parameters requiring no tuning because they are commonly selected in the literature. In the top of Table D.2 and D.1, we write down the these parameters and explain the reason for choosing a particular value.

Tunable Hyper-parameters. In the bottom of Table D.2 and D.1, we show our choice of hyper-parameters for ImageNet and CIFAR-10/100, respectively. This includes

- `optimizer`. We tested a momentum-SGD (`sgd+momentum`) and an Adam optimizer (`adam`). We found that momentum-SGD is more stable in fine-tuning the ViT from a pre-trained checkpoint.
- `base_lr`, `warm_up_iterations` and `decay_type`. We linearly increase the learning rate from 0 to the `base_lr` during `warm_up_iterations`. After warm-up, we gradually schedule the learning rate based on the `decay_type`. We experiment with a multi-step and a cosine decay and find no apparent difference between these two schedulers. In the end, we choose cosine because it has less hyper-parameters to choose compared to the multi-step one.
- `l2_reg`. On ImageNet, we find ℓ_2 regularization with a penalty of 0.0001 helps the baseline TRADES(base). On CIFAR-10/100, we find that ℓ_2 regularization may not be necessary when using DDPM data.
- `cutmix`. In both cases we do not find the cut mix augmentation (Yun et al., 2019) help to improve the results.
- `data_range`. On CIFAR10/100, we find that centered data, i.e. normalizing the data to have 0 mean and (close to) 1 standard deviation, provides better results than scaling the images to [-1, 1] (the range of data used by the pre-trained checkpoint). We simply subtract the CIFAR images from the average of per-channel mean (0.47) and divide it with the average of per-channel standard deviation (0.25). For ImageNet, we still use [-1, 1] as the data range. Notice that ball needs to be re-scaled for both data ranges describe above. For example, when reporting results on $\epsilon = 0.031$ and using centered data, we need to use $\epsilon/0.25$ as the actual noise bound passed to the attacker. When normalizing the data to $[-1, 1]$, we need to pass $\epsilon/0.5$ to the attacker.
- `patch_size`. The size of the image patch of the input sequence to ViT. 16×16 is the standard size of pre-trained ViT models on ImageNet21K. Thus, when fine-tuning on ImageNet, we use 16×16. CIFAR images are a lot smaller compared to ImageNet images. As a result, we use a smaller patch

Fine-tuning Hyper-parameters (CIFAR-10/100, ViT-L16)

Defense	Hyper-parameter	Range	Final Choice (CIFAR-10)	Final Choice (CIFAR-100)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(`TrH`)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.00001	0.01
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(`TrH`), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'constant'
	γ	-	0.001	0.001

Fine-tuning Hyper-parameters (CIFAR-10/100, Hybrid-L16)

Defense	Hyper-parameter	Range	Final Choice (CIFAR-10)	Final Choice (CIFAR-100)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(`TrH`)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(`TrH`), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001

Table D.3: Hyper-parameters used in each defense and the values used to reproduce CIFAR10/100 results in Table. 4.1.

size of 4×4 to produce more patches. Using a smaller patch size requires some modifications to ViT architecture and we discuss this in detail in Appendix D.5.

- `downsample` and `patch_stride`. These are particular to CIFAR images. Please refer to Appendix D.5 for detail.

D.3 Hyper-parameters for Specific Methods

We fine-tune ViTs after common hyper-parameters are locked after pre-tuning. For all methods, we take 10 PGD steps on CIFAR-10/100 and 7 steps for ImageNet during the training. We report the best results for each method after trying different sets of hyper-parameters. This usually involves method-specific parameters. We elaborate what hyper-paramter is tuned as follows and report the final values used in the experiments in Table D.3 (CIFAR-10/100) and D.4 (ImageNet).

Fine-tuning Hyper-parameters (ImageNet, ViT-B16)

Defense	Hyper-parameter	Range	For ℓ_∞	For ℓ_2
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'linear'
	γ	-	0.0001	0.0001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.00005	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'linear'	'linear'
	γ	-	0.0001	0.0001

Fine-tuning Hyper-parameters (ImageNet, ViT-L16)

Defense	Hyper-parameter	Range	For ℓ_∞	For ℓ_2
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'linear'
	γ	-	0.0001	0.0001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.0001	0.0001

Table D.4: Hyper-parameters used in each defense and the values used to reproduce ImageNet results in Table. 4.1.

- **base**: we use $\lambda_t = 6$ for TRADES training and λ_t is consistent across all experiments.
- **SWA**: we use $\alpha = 0.995$ so that $\theta_{\texttt{avg}} \leftarrow 0.995 * \theta_{\texttt{avg}} + 0.005 * \theta^{(t+1)}$ as this value is used in the literature (Gowal et al., 2021). Therefore, there is no tuning in SWA.
- **S2O**: The only parameter that requires tuning is the penalty α of the second-order statistic in Eq 4.2. In the authors' implementation, we find $1 - \alpha$ is used to balance the regularization with the robust loss (AT or TRADES). These hyper-parameters are hard-coded in the latest commit f2d037b[1] so we directly use their choice of hyper-parameters. Namely, for AT loss, the finally loss in S2O training is set to

$$0.9 * AT\_loss + 0.1 * S2O\_loss$$

and for TRADES training the final loss is

$$0.7 * \frac{1}{m}\sum_i^m \mathrm{CE}((x_i, y_i), F_\theta) + 0.3 * S2O\_loss + \frac{\lambda_t}{m}\sum_i^m \max_{||\epsilon_i||_p \leq \delta} \mathrm{KL}((x_i, \epsilon_i), F_\theta).$$

[1] https://github.com/Alexkael/S2O/tree/f2d037b9611f73227834118250992 51f7978f54e

- **AWP**: The two hyper-parameters in AWP that requires tuning are ψ and δ_{awp}, where ψ is the function to measure the noise added to the weights and δ_{awp} is the noise budget. We follow the choice made by Wu et al. (Wu et al., 2020) to choose ψ as the layer-wise ℓ_2 norm function so that the noise added to the weights in each layer should no greater than the ℓ_2 norm of the weights multiplied by δ_{awp}. Namely, suppose that $\xi^{(i)}$ is the noise added to a weight matrix $W^{(i)}$ at layer i, then we project $\xi^{(i)}$ such that
$$\frac{||\xi^{(i)} + W^{(i)}||_2}{||W^{(i)}||_2} \leq \delta_{awp}.$$
For the choice of δ_{awp}, we sweep δ_{awp} over the interval {0.0001, 0.0005, 0.001, 0.005, 0.01}.

- **TrH**: The two hyper-parameters in TrH that requires tuning are the ℓ_2 weight penalty γ and the TrH penalty λ. For γ, we find 0.001 as a reasonable choice for CIFAR-10/100 and 0.0001 as a reasonable choice for ImageNet. For λ, we sweep the interval {0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}. Furthermore, we also consider three types of schedulers: 'constant', 'linear', 'multistep(0.1-0.5:0.1)' for λ scheduling. Intuitively, a strong TrH regularization at the very beginning may lead the model to a flat highland instead of a flat minimum where we get a degenerated model. Ramping up λ to the chosen value allows the model to focus more on accuracy and robustness at the early stage. In practice, we find that CIFAR10/100 is not sensitive to the choice of the λ schedulers; however, ImageNet favors 'linear' or 'multistep' schedulers over the 'constant' λ.
  - 'constant'. No λ scheduling.
  - 'linear'. We ramp up λ from 0 to the chosen value from iteration 1 to the end.
  - 'multistep(0.1-0.5:0.1)'. We use 0.01 * λ before finishing 10% of the total iterations. We use 0.1 * λ after finishing 10% and before 50% of the total iterations. After finishing 50% of iterations, we use λ.

D.4 Additional Results

We include the results on CIFAR-10/100 with ViT-B16 in Table D.5 and ImageNet with Hybrid-L16 in Table D.6 of the appendix. These additional results are consistent with what we have found in Section **??** of the paper: in CIFAR-10/100, TrH is consistently among the top or silver results; in ImageNet, TrH has significantly advantages over the other baseline methods.

D.5 ViT Architecture

We describe the architectures of Vision Transformers used in our experiments in Section 4.5.

$\ell_\infty(\epsilon = 8/255)$	ViT-B16			
SE= ±0.5%	CIFAR-10		CIFAR-100	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(base)	87.5	60.3	60.0	30.4
AT(SWA)	86.9	60.4	64.1	**33.7**
AT(S2O)	86.8	60.4	63.2	31.5
AT(AWP)	87.3	61.3	61.5	32.2
AT(TrH)	88.4	**61.5**	65.0	33.0
TRADES(base)	85.4	60.8	58.6	30.5
TRADES(SWA)	85.6	60.6	62.7	**33.0**
TRADES(S2O)	85.9	**61.1**	63.5	31.5
TRADES(AWP)	84.7	60.1	60.8	32.2
TRADES(TrH)	85.8	**61.1**	63.8	**33.0**

Table D.5: Additional results for CIFAR-10/100 using ViT-B16. Clean: % of Top-1 correct predictions. ERA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) Stark & University of California (2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (n as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

ImageNet	Hybrid-L16			
SE= ±0.2%	$\ell_\infty(\epsilon = 4/255)$		$\ell_2(\epsilon = 3.0)$	
Defense	Clean(%)	ERA(%)	Clean (%)	ERA(%)
AT(base)	72.6	40.7	72.2	40.6
AT(SWA)	72.7	40.4	72.7	40.5
AT(S2O)	72.8	43.6	72.3	40.9
AT(AWP)	67.7	39.4	67.9	40.3
AT(TrH)	75.0	**46.2**	74.4	**45.9**
TRADES(base)	79.5	37.6	78.0	38.6
TRADES(SWA)	72.9	40.9	71.3	40.8
TRADES(S2O)	73.8	41.3	72.2	41.2
TRADES(AWP)	66.4	38.8	65.3	40.9
TRADES(TrH)	75.9	**45.7**	73.3	**45.6**

Table D.6: Addtional Results for ImageNet using Hybird-L16. Clean: % of Top-1 correct predictions. ERA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) Stark & University of California (2005) = $\sqrt{0.5 * (1 - 0.5)/n}$ (n as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

- ViT-B16 includes 12 transformer layers with hidden size 768, MLP layer size 3072 and 12 heads in the multi-head attention.
- ViT-L16 includes 24 transformer layers with hidden size 1024, MLP layer size 4096 and 16 heads in the multi-head attention.
- Hybird-L16 is a hybrid model of a ResNet-50 and a ViT-L16 model. The patches fed into ViT are feature

representations encoded by a ResNet-50 and projected to the Transformer dimensions (Dosovitskiy et al., 2021). Dissimilar to a standard ResNet-50 feature encoder, Dosovitskiy et al. (Dosovitskiy et al., 2021) replaced Batch Normalization with Group Normalization (Wu & He, 2019) and use standardized Convolution (Qiao et al., 2019). Moreover, Dosovitskiy et al. (Dosovitskiy et al., 2021) removed stage 4, placed the same number of layers in stage 3 (keeping the total number of layers), and took the output of this extended stage 3 as the input of ViT.

The pretrained ViT-B16 and ViT-L16 checkpoints use patch size 16×16. However, since the images in CIFAR10/100 are of size 32×32, there will be only 4 patches when using the original patch size. Therefore, we downsample the kernel of the first convolution to produce image patches of 4×4. Among different ways of downsampling we find the `cubic` interpolation gives the best results, which is the one chosen in pre-tuning as discussed in Appendix D.2. Indeed, Mahmood et al. (Mahmood et al., 2021) empirically finds that such down-sampling provides better results for CIFAR10 images. Furthermore, in generating patches, we also use a stride of 2 instead of 4, because we find using overlapped patches increases the sequence length and results in better performance.

Bibliography

doi: 10.1007/s11023-020-09539-2. URL https://doi.org/10.1007year=2020,month={sep}, publisher={SpringerScienceandBusiness},volume={30},number={3},pages={411--437}, author={IasonGabriel},title={ArtificialIntelligence,Values,andAlignment},journal= {MindsandMachines}. 1

Github - deepmind-research/adversarial_robustness. URL https://github.com/deepmind/deepmind-research/tree/master/adversarial_robustness. 53

Github - google-research/vision_transformer. URL https://github.com/google-research/vision_transformer. 53

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps, 2018. 23, 24

Alain, G., Roux, N. L., and Manzagol, P.-A. Negative eigenvalues of the hessian in deep neural networks, 2018. URL https://openreview.net/forum?id=S1iiddyDG. 47

Alquier, P. User-friendly introduction to pac-bayes bounds, 2021. 7, 42, 45

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018a. 13, 24, 25

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018b. 18

Anil, C., Lucas, J., and Gross, R. Sorting out Lipschitz function approximation. In *International Conference on Machine Learning (ICML)*, 2019. 64

Araujo, A., Havens, A. J., Delattre, B., Allauzen, A., and Hu, B. A unified algebraic perspective on lipschitz neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. 70, 71, 72

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2), apr 2011. ISSN 0004-5411. doi: 10.1145/1944345.1944349. 47

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. 2

Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations (ICLR)*, 2020. 75

Beyer, L., Hénaff, O. J., Kolesnikov, A., Zhai, X., and Oord, A. v. d. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020. 76

Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., and Samek, W. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*, pp. 63–71. Springer, 2016. 4, 15, 89

Black, E., Wang, Z., and Fredrikson, M. Consistent counterfactuals for deep models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=St6eyiTEHnG. 89

BOHNENBLUST, H. F., BROWN, G. W., DRESHER, M., GALE, D., KARLIN, S., KUHN, H. W., MCKINSEY, J. C. C., NASH, J. F., NEUMANN, J. V., SHAPLEY, L. S., SHERMAN, S., SNOW, R. N., TUCKER, A. W., and WEYL, H. *Contributions to the Theory of Games (AM-24), Volume I*. Princeton University Press, 1952. ISBN 9780691079349. URL http://www.jstor.org/stable/j.ctt1b9rzq1. 4, 15, 16, 89

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., Khlaaf, H., Yang, J., Toner, H., Fong, R., et al. Toward trustworthy ai development: mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020. 1

Bubeck, S. and Sellke, M. A universal law of robustness via isoperimetry. In *NIPS*, 2021. 75

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017a. 21

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017b. 34

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017c. 2

Carlini, N., Tramer, F., Kolter, J. Z., et al. (certified!!) adversarial robustness for free! *arXiv preprint arXiv:2206.10550*, 2022. 74

Catoni, O. A pac-bayesian approach to adaptive classification. 2004. 7, 42, 45

Chalasani, P., Chen, J., Chowdhury, A. R., Wu, X., and Jha, S. Concise explanations of neural networks using adversarial training. In *International Conference on Machine Learning*, pp. 1383–1391. PMLR, 2020. 38

Chen, J., Wu, X., Rastogi, V., Liang, Y., and Jha, S. Robust attribution regularization. In *Advances in Neural Information Processing Systems*, 2019. 38

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017. 1, 2

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019a. 8, 9, 59, 60, 74, 75

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019b. 39

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020a. 21, 34, 53, 91

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020b. 2, 7, 35

Croce, F., Andriushchenko, M., and Hein, M. Provable robustness of ReLU networks via maximization of linear regions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. 8, 59

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020. 94

Das, S., Singh, A., Chatterjee, S., Bhattacharya, S., and Bhattacharya, S. Finding high-value training data subset through differentiable convex programming. In *ECML/PKDD*, 2021. 89

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2, 5, 26

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 94

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkeryxBtPB. 58

Ding, N., Chen, X., Levinboim, T., Goodman, S., and Soricut, R. Bridging the gap between practice and pac-bayes theory in few-shot meta-learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29506–29516. Curran Associates, Inc., 2021. 46

Ding, N., Chen, X., Levinboim, T., Changpinyo, B., and Soricut, R. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *ECCV*, 2022. 45, 46, 57

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021*, 2020. 5, 30, 31, 95

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy. 53, 121

Dozat, T. Incorporating nesterov momentum into adam. 2016. 94

Du, A., Chen, B., Chin, T.-J., Law, Y. W., Sasdelli, M., Rajasegaran, R., and Campbell, D. Physical adversarial attacks on an aerial imagery object detector. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1796–1806, 2022. 34

Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pp. 2782–2792. PMLR, 2020. 80

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness. 37

Erion, G., Janizek, J. D., Sturmfels, P., Lundberg, S. M., and Lee, S.-I. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature machine intelligence*, 2021. 4, 15, 24, 25, 89

Farnia, F., Zhang, J., and Tse, D. Generalizable adversarial training via spectral normalization. In *ICLR*, 2019. 63

Feng, L., Ahmed, M. O., Hajimirsadeghi, H., and Abdi, A. H. Towards better selective classification. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5gDz_yTcst. 80

Fischer, M., Baader, M., and Vechev, M. Scalable certified segmentation via randomized smoothing. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3340–3351. PMLR, 18–24 Jul 2021. 76

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3449–3457, 2017. 4, 15, 89

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM. 44, 49, 57

Fromherz, A., Leino, K., Fredrikson, M., Parno, B., and Păsăreanu, C. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations (ICLR)*, 2021a. 19, 21

Fromherz, A., Leino, K., Fredrikson, M., Parno, B., and Păsăreanu, C. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations (ICLR)*, 2021b. 8, 9, 59, 60

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016. 80

Geifman, Y. and El-Yaniv, R. Selectivenet: A deep neural network with an integrated reject option. In *International conference on machine learning*, pp. 2151–2159. PMLR, 2019. 80

Germain, P., Lacasse, A., Laviolette, F., and Marchand, M. Pac-bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 353–360, New

York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553419. URL https://doi.org/10.1145/1553374.1553419. 7, 42, 45, 46

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. Pac-bayesian theory meets bayesian inference. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016a. URL https://proceedings.neurips.cc/paper/2016/file/84d2004bf28a2095230e8e14993d398d-Paper.pdf. 7, 45

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. Pac-bayesian theory meets bayesian inference. volume 29, 2016b. 46

Ghalebikesabi, S., Ter-Minassian, L., Diaz-Ordaz, K., and Holmes, C. C. On locality of local explanation models. *arxiv*, abs/2106.14648, 2021. 4, 15, 89

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015a. 19

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015b. 2, 34, 44

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015c. 5

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples, 2015d. 7

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, Feb 2021. 63, 64

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. Improving robustness using generated data. In *NeurIPS*, 2021. 7, 8, 43, 44, 45, 52, 53, 58, 118

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 94

Han, X. and Tsvetkov, Y. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. *arXiv preprint arXiv:2110.03212*, 2021. 89

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015. 23, 31, 64

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 63

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021. 2

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 52

Howard, J. imagenette. URL https://github.com/fastai/imagenette/. 31, 35

Hu, K., Zou, A., Wang, Z., Leino, K., and Fredrikson, M. Scaling in depth: Unlocking robustness certification on imagenet, 2023. 65, 70, 94

Huang, L., Zhang, C., and Zhang, H. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33:19365–19376, 2020. 80

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=FTt28RYj5Pc. 52

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. In *NIPS*, 2021b. 8, 59, 61, 63, 70, 71, 74

Huster, T., Chiang, C.-Y. J., and Chadha, R. Limitations of the lipschitz constant as a defense against adversarial examples. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2018. 63, 73

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 5, 31

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019. 33

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pp. 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018. 7, 8, 44, 45, 52, 53

Jeong, J., Park, S., Kim, M., Lee, H.-C., Kim, D.-G., and Shin, J. Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness. *Advances in Neural Information Processing Systems*, 34: 30153–30168, 2021. 74

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D. X., and Spanos, C. J. Towards efficient data valuation based on the shapley value. In *International Conference on Artificial Intelligence and Statistics*, 2019. 89

Jia, Z. and Su, H. Information-theoretic local minima characterization and regularization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4773–4783. PMLR, 2020. 57

Jin, G., Yi, X., Huang, W., Schewe, S., and Huang, X. Enhancing adversarial training with second-order statistics of weights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15273–15283, June 2022. 7, 8, 45, 46, 52, 53

Jordan, M., Lewis, J., and Dimakis, A. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In *NeurIPS*, 2019a. 19, 21

Jordan, M., Lewis, J., and Dimakis, A. G. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In *Neural Information Processing Systems (NIPS)*, 2019b. 8, 59

Jouppi, N. P., Young, C., Patil, N., Patterson, D. A., Agrawal, G., Bajwa, R. S., Bates, S., Bhatia, S., Boden, N. J., Borchers, A., Boyle, R., luc Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R. B., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K. A., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. In-datacenter performance analysis of a tensor processing unit. *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017. 53

Ju, H., Li, D., and Zhang, H. R. Robust fine-tuning of deep neural networks with hessian-based generalization guarantees. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 10431–10461. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ju22a.html. 57

Kannan, H., Kurakin, A., and Goodfellow, I. J. Adversarial logit pairing. *ArXiv*, abs/1803.06373, 2018. 111

Karimi, A.-H., von Kügelgen, J., Schölkopf, B., and Valera, I. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *arXiv preprint arXiv:2006.06831*, 2020. 89

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer-Aided Verification (CAV)*, 2017. 9, 60

Keane, M. T. and Smyth, B. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *International Conference on Case-Based Reasoning*, pp. 163–178. Springer, 2020. 89

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018. 89

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017. 89

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., and Reblitz-Richardson, O. Captum: A unified and generic model interpretability library for pytorch, 2020. 92

Kolter, J. Z. and Wong, E. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018. 21

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`. 37

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 30

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. *ArXiv*, abs/1607.02533, 2016. 34

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. 80

Lee, M. and Kolter, Z. On physical adversarial patches for object detection. *arXiv preprint arXiv:1906.11897*, 2019. 34

Lee, S., Lee, J., and Park, S. Lipschitz-certifiable training with a tight outer bound. *Advances in Neural Information Processing Systems*, 33, 2020a. 21, 70, 71

Lee, S., Lee, J., and Park, S. Lipschitz-certifiable training with a tight outer bound. In *Neural Information Processing Systems (NIPS)*, 2020b. 8, 59

Leino, K. Identifying, analyzing, and addressing weaknesses in deep networks: Foundations for conceptually sound neural networks, 2022. 69

Leino, K. Limitations of piecewise linearity for efficient robustness certification. 2023. 75

Leino, K. and Fredrikson, M. Relaxing local robustness. In *Neural Information Processing Systems (NIPS)*, 2021. 76

Leino, K., Sen, S., Datta, A., Fredrikson, M., and Li, L. Influence-directed explanations for deep convolutional networks. In *2018 IEEE International Test Conference (ITC)*, pp. 1–8. IEEE, 2018. 4, 13, 15, 89

Leino, K., Rshih32, Gopinath, D., Anupam Datta, Shayak Sen, MacKlinkachorn, Kaiji Lu, and Zifan Wang. truera/trulens: Trulens, 2021a. URL `https://zenodo.org/record/4495856`. xi, 33

Leino, K., Wang, Z., and Fredrikson, M. Globally-robust neural networks. In *International Conference on Machine Learning*, pp. 6212–6222. PMLR, 2021b. 52, 70, 85, 94

Leino, K., Wang, Z., and Fredrikson, M. Globally-robust neural networks, 2021c. 21, 34

Leino, K., Wang, Z., and Fredrikson, M. Globally-robust neural networks. In *International Conference on Machine Learning (ICML)*, 2021d. 8, 9, 59, 60, 61, 72

Li, Q., Haque, S., Anil, C., Lucas, J., Grosse, R., and Jacobsen, J.-H. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Conference on Neural Information Processing Systems*, 2019a. 8, 59

Li, Q., Haque, S., Anil, C., Lucas, J., Grosse, R. B., and Jacobsen, J.-H. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *Neural Information Processing Systems (NIPS)*, 2019b. 8, 59, 70, 71

Lin, J., Zhang, A., Lécuyer, M., Li, J., Panda, A., and Sen, S. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pp. 13468–13504. PMLR, 2022. 89

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 94

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777, 2017. 4, 15, 90

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019. 80

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 7, 43, 44, 53, 91

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018a. 2, 80

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018b. 21, 25, 34

Mahajan, D., Tan, C., and Sharma, A. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019. 89

Mahmood, K., Mahmood, R., and van Dijk, M. On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7838–7847, October 2021. 53, 115, 121

Mao, X., Chen, Y., Li, X., Qi, G., Duan, R., Zhang, R., and Xue, H. Easyrobust: A comprehensive and easy-to-use toolkit for robust computer vision. `https://github.com/alibaba/easyrobust`, 2022. 35

McAllester, D. A. Pac-bayesian model averaging. In *COLT '99*, 1999. 7, 42, 45

Meunier, L., Delattre, B. J., Araujo, A., and Allauzen, A. A dynamical system perspective for Lipschitz neural networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15484–15500. PMLR, 17–23 Jul 2022. 70, 71

Mothilal, R. K., Sharma, A., and Tan, C. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617, 2020. 89

Muggleton, S. and Chater, N. *Human-Like Machine Intelligence*. Oxford University Press, 07 2021. ISBN 9780198862536. doi: 10.1093/oso/9780198862536.001.0001. URL https://doi.org/10.1093/oso/9780198862536.001.0001. 1

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017. 45

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I. M., and Edwards, B. Adversarial robustness toolbox v1.0.0, 2019. 7, 8, 91

Northcutt, C. G., Athalye, A., and Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021. 76

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. 1

Pan, D., Li, X., and Zhu, D. Explaining deep neural network models with adversarial gradient integration. In *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2021. 4, 15, 25, 90

Pang, T., Lin, M., Yang, X., Zhu, J., and Yan, S. Robustness and accuracy could be reconcilable by (proper) definition. In *ICML*, 2022. 58

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 94

Pawelczyk, M., Broelemann, K., and Kasneci, G. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, pp. 3126–3132, 2020. 89

Petsiuk, V., Das, A., and Saenko, K. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018. 4, 15, 90

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., and Flach, P. Face: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 344–350, 2020. 89

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020. 89

Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019. 121

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. URL http://arxiv.org/abs/1707.04131. 91

Rauber, J., Zimmermann, R., Bethge, M., and Brendel, W. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020. doi: 10.21105/joss.02607. URL https://doi.org/10.21105/joss.02607. 91

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135–1144, 2016. 4, 15, 90

Rice, L., Wong, E., and Kolter, J. Z. Overfitting in adversarially robust deep learning. In *ICML*, 2020. 7, 44

Roth, K., Kilcher, Y., and Hofmann, T. Adversarial training is a form of data-dependent operator norm regularization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14973–14985. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ab7314887865c4265e896c6e209d1cd6-Paper.pdf. 52

Rothfuss, J., Fortuin, V., Josifoski, M., and Krause, A. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pp. 9116–9126. PMLR, 2021. 46

Russell, S. J. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010. 1

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019. 74

Salman, H., Sun, M., Yang, G., Kapoor, A., and Kolter, J. Z. Denoised smoothing: A provable defense for pretrained classifiers. *Advances in Neural Information Processing Systems*, 33:21945–21957, 2020. 74

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022. 89

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128:336–359, 2019. 4, 15, 24, 90

Senechal, M. Mathematical structures: spatial tessellations. concepts and applications of voronoi diagrams. atsuyuki okabe, barry boots, and kokichi sugihara. wiley, new york, 1992. xii, 532 pp., illus. $89.95. wiley series in probability and mathematical statistics. *Science*, 260(5111):1170–1173, 1993. doi: 10.1126/science.260.5111.1170. URL https://www.science.org/doi/abs/10.1126/science.260.5111.1170. 82

Shao, J., Hu, K., Wang, C., Xue, X., and Raj, B. Is normalization indispensable for training deep neural network? *Advances in Neural Information Processing Systems*, 33:13434–13444, 2020. 95

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. 34

Shawe-Taylor, J. and Williamson, R. C. A pac analysis of a bayesian estimator. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, COLT '97, pp. 2–9, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918916. doi: 10.1145/267460.267466. URL https://doi.org/10.1145/267460.267466. 7, 42, 45

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017. 4, 15, 24, 25, 90

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015. 31

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013. 4, 15, 16, 30, 90

Singla, S. and Feizi, S. Skew orthogonal convolutions. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9756–9766. PMLR, 18–24 Jul 2021. 63, 70, 75

Singla, S., Singla, S., and Feizi, S. Improved deterministic l2 robustness on cifar-10 and cifar-100. In *ICLR*, 2022. xii, 8, 59, 64, 65, 70, 71, 96

Sinha, A., Namkoong, H., and Duchi, J. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations (ICLR)*, 2018. 9, 60

Sinha, A., Namkoong, H., Volpi, R., and Duchi, J. Certifying some distributional robustness with principled adversarial training, 2020. 21

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise, 2017. 4, 15, 25, 90

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net, 2014. 24

Stark, P. and University of California, B. D. o. S. *SticiGui: Statistics Tools for Internet and Classroom Instruction*. Department of Statistics. University of California, Berkeley, 2005. URL https://books.google.com/books?id=qwpsAQAACAAJ. viii, x, 54, 56, 71, 120

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017. 4, 15, 17, 22, 90

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019a. 8, 59

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019b. 21

Trockman, A. and Kolter, J. Z. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations (ICLR)*, 2021. 8, 9, 59, 60, 70, 71, 75

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Neural Information Processing Systems*, 2018. 69

Ustun, B., Spangher, A., and Liu, Y. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 10–19, 2019. 89

Van Looveren, A. and Klaise, J. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019. 89

Vasudevan, V., Caine, B., Gontijo-Lopes, R., Fridovich-Keil, S., and Roelofs, R. When does dough become a bagel? analyzing the remaining mistakes on imagenet. *arXiv preprint arXiv:2205.04596*, 2022. 76

Viallard, P., VIDOT, E. G., Habrard, A., and Morvant, E. A pac-bayes analysis of adversarial robustness. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14421–14433. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/78e8dffe65a2898eef68a33b8db35b78-Paper.pdf. 57

Wang, H., Wu, X., Yin, P., and Xing, E. P. High-frequency component helps explain the generalization of convolutional neural networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8681–8691, 2019. 33

Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., and Hu, X. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 24–25, 2020a. 4, 15, 24, 27, 90

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=rklOg6EFwS. 58, 113

Wang, Z., Wang, H., Ramkumar, S., Mardziel, P., Fredrikson, M., and Datta, A. Smoothed geometry for robust attribution. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, 2020c. 38, 39

Wang, Z., Fredrikson, M., and Datta, A. Robust models are more interpretable because attributions look normal. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22625–22651. PMLR, 17–23 Jul 2022. 4, 15, 28, 90

Wang, Z., Yao, Y., Zhang, C., Zhang, H., Kang, Y., Joe-Wong, C., Fredrikson, M., and Datta, A., 2023. URL https://arxiv.org/abs/2205.11850. 17, 22

Wen, K., Ma, T., and Li, Z. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022. 49

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning (ICML)*, 2018a. 9, 73

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C., Boning, D., Dhillon, I., and Daniel, L. Towards fast computation of certified robustness for relu networks. In *ICML*, 2018b. 21

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018a. 8, 59, 70

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018b. 60, 70, 71

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJx040EFvH. 7, 44

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020. 7, 8, 44, 45, 46, 52, 53, 57, 119

Wu, Y. and He, K. Group normalization. *International Journal of Computer Vision*, 128:742–755, 2019. 121

Xu, X., Li, L., and Li, B. Lot: Layer-wise orthogonal training on improving l2 certified robustness. *arXiv preprint arXiv:2210.11620*, 2022. 70, 71

Yang, L., Kenny, E. M., Ng, T. L. J., Yang, Y., Smyth, B., and Dong, R. Generating plausible counterfactual explanations for deep transformers in financial text classification. *arXiv preprint arXiv:2010.12512*, 2020a. 89

Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R. R., and Chaudhuri, K. A closer look at accuracy vs. robustness. In *Neural Information Processing Systems (NIPS)*, 2020b. 61, 75

Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018. 89

Yeh, C.-K., Hsieh, C.-Y., Suggala, A. S., Inouye, D. I., and Ravikumar, P. On the (in) fidelity and sensitivity for explanations. In *Advances in Neural Information Processing Systems*, 2019. 22, 23, 25

Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020. 89

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019. 7, 44, 116

Yun, S., Oh, S. J., Heo, B., Han, D., Choe, J., and Chun, S. Re-labeling imagenet: From single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2340–2350, June 2021. 76

Zecchin, M., Park, S., Simeone, O., Kountouris, M., and Gesbert, D. Robust pacm: Training ensemble models under model misspecification and outliers. *ArXiv*, abs/2203.01859, 2022. 57

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 7, 44

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *Neural Information Processing Systems (NIPS)*. 2018. 9

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. *ICLR*, 2019a. 95

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019b. x, 8, 43, 66

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019c. 7

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations (ICLR)*, 2020. 75

Zhang, M., Lucas, J., Ba, J., and Hinton, G. E. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019d. 94

Zhou, B., Khosla, A., A., L., Oliva, A., and Torralba, A. Learning Deep Features for Discriminative Localization. *CVPR*, 2016. 24

Ziyin, L., Wang, Z., Liang, P. P., Salakhutdinov, R., Morency, L.-P., and Ueda, M. Deep gamblers: Learning to abstain with portfolio theory. *arXiv preprint arXiv:1907.00208*, 2019. 80