On the Stability of Neural Networks in Deep Learning

Blaise Delattre

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Université Paris-Dauphine-PSL

On the Stability of Neural Networks in Deep Learning

Soutenue par
Blaise Delattre
Le 20 Juin 2025

École doctorale n°543
École Doctorale SDOSE

Spécialité
Informatique

Composition du jury :

Francis BACH	Directeur de recherche CRNS, ENS-PSL	*Président*
Rémi GRIBONVAL	Directeur de recherche, INRIA Lyon	*Rapporteur*
Audrey REPETTI	Associate Professor, Maxwell Institute for Mathematical Sciences Edinburgh	*Rapporteure*
Claire BOYER	Professeure des universités Institut de Mathématiques d'Orsay, Université Paris-Saclay	*Examinatrice*
Franck MAMALET	Senior Expert in Artificial Intelligence, IRT St-Exupery	*Examniateur*
Quentin BARTHELEMY	Chercheur, Foxstream Lyon	*Encadrant de thèse*
Alexandre ALLAUZEN	Professeur, Paris-Dauphine-PSL	*Directeur de thèse*

Dauphine | PSL
UNIVERSITÉ PARIS

Funding and Grants

This doctoral research was conducted within the Machine Intelligence and LEarning Systems (MILES) team of the LAMSADE research lab at Université Paris-Dauphine / Université PSL, between April 2022 and March 2025, as part of a CIFRE collaboration with the company Foxstream. It was supported by the French National Research Agency (ANR) through the SPEED project (ANR-20-CE23-0025) and by the French Government via the France 2030 program (ANR-23-PEIA-0008, SHARP). Part of this work was performed using high-performance computing resources provided by GENCI-IDRIS (Grant 2023-AD011014214R1).

Remerciements

Je souhaite exprimer ma profonde reconnaissance à l'ensemble des membres du jury pour l'honneur qu'ils m'ont fait en acceptant d'évaluer ce travail. Je remercie Francis Bach, Président du jury, ainsi que Claire Boyer et Franck Mamalet, examinateurs, pour l'attention portée à cette soutenance et pour la pertinence de leurs remarques et questions. J'adresse une gratitude toute spéciale à Rémi Gribonval et à Audrey Repetti, rapporteurs de cette thèse, pour la rigueur et la précision de leur lecture, ainsi que pour la richesse et la profondeur de leurs commentaires. Leurs observations détaillées ont grandement contribué à améliorer la qualité de ce manuscrit.

J'exprime aussi mes remerciements les plus sincères à mes encadrants. Je remercie tout particulièrement Quentin Barthélemy pour son encadrement soutenu et bienveillant, pour sa disponibilité, ainsi que pour m'avoir fait découvrir le monde de la recherche. Je suis également profondément reconnaissant à Alexandre Allauzen pour son regard avisé, mais aussi pour sa bienveillance et sa bonne humeur, qui ont contribué à rendre ces années de thèse particulièrement stimulantes et enrichissantes. Je tiens enfin à remercier Alexandre Araujo pour son accompagnement précieux au début de ma thèse et pour l'aide qu'il m'a apportée.

Je suis reconnaissant à l'ensemble des membres de l'entreprise Foxstream pour leur accueil chaleureux et leur sympathie tout au long de cette collaboration. Je souhaite aussi remercier l'équipe MILES, que j'ai eu beaucoup de plaisir à rencontrer et à côtoyer au quotidien. Discussions, collaborations et moments de vie partagés ont largement contribué à rendre ces années riches et agréables. Je pense en particulier à Yann, Benjamin et Jamal, mais aussi aux doctorants et post-doctorants Alexandre, Paul, Florian, Erwan et Lucas, avec qui j'ai eu la chance de collaborer et de partager de nombreux moments de camaraderie.

Enfin, je souhaite remercier mes amis, ma famille et ma compagne pour tout ce qu'ils m'apportent jour après jour et plus particulièrement durant ces années de thèse. Je pense en particulier à mes parents, qui m'ont toujours encouragé dans ce projet depuis le début, et à mon père, qui m'a transmis le goût de la recherche et avec qui j'ai aujourd'hui le plaisir de le partager.

Contents

1 Introduction

Contents

1.1 Success of deep learning

Deep learning has driven groundbreaking progress across various domains, fundamentally transforming modern technology. In computer vision, its impact has been profound, beginning with early applications in digit recognition and image classification (LeCun et al., 1998). The field reached a milestone with the success of AlexNet, which achieved a dramatic breakthrough on ImageNet by significantly outperforming traditional methods using deep convolutional neural networks (CNNs) (Krizhevsky et al., 2012). This success has inspired advancements in diverse fields, from healthcare and biology to natural language processing and robotics, showcasing deep learning's transformative potential.

In the medical domain, deep learning has revolutionized imaging, with U-Net (Ronneberger et al., 2015) enabling precise segmentation of biomedical structures, advancing cancer detection, and automating diagnostic tasks (Falk et al., 2019). Similarly, AlphaGo's victory against world champions in the board game Go demonstrated deep learning's capacity for strategic decision-making through reinforcement learning (Silver et al., 2016). Biology has also witnessed transformative advancements, most notably in protein structure prediction. AlphaFold (Jumper et al., 2021) resolved a decades-old challenge, achieving near-experimental accuracy in predicting protein structures and paving the way for advances in drug discovery and molecular biology. Generative models have expanded the boundaries of creativity and utility

across numerous domains. GANs (Goodfellow et al., 2014) have set the standard for high-quality image synthesis, enabling applications such as photorealistic image generation, artistic creation, and deepfake technology. Variational Autoencoders (VAEs) (Kingma and Welling, 2014) have advanced probabilistic modeling and latent space representation, while diffusion models (Ho et al., 2020) have emerged as state-of-the-art for generating realistic and diverse outputs in fields like molecular design and scientific simulations. CNN-based architectures are fundamental to computer vision. Models like YOLO (Redmon et al., 2016; al., 2020) are essential for real-time object detection, combining accuracy with low latency for tasks such as autonomous driving. Modern CNNs such as ConvNeXt (Liu et al., 2022) continue to deliver state-of-the-art performance with computational efficiency. Vision Transformers (ViTs) (Dosovitskiy et al., 2021) have further extended deep learning's capabilities by leveraging self-attention mechanisms (Vaswani et al., 2017) to excel in large-scale image classification tasks. In natural language processing, attention has enabled significant advancements, replacing recurrent neural networks (Hochreiter and Schmidhuber, 1997b; Cho et al., 2014) as the dominant architecture for sequence modeling, with early attention mechanisms first introduced in neural machine translation (Bahdanau et al., 2015) and later generalized to self-attention (Vaswani et al., 2017). This shift has led to the development of large language models (LLMs) that have driven remarkable breakthroughs. BERT (Devlin et al., 2019) introduced bidirectional pre-training, enabling context-aware embeddings that revolutionized tasks such as question answering and sentiment analysis. Generative models like GPT (Brown et al., 2020) have scaled autoregressive training to generate coherent and contextually relevant text, transforming applications such as conversational AI, machine translation, and content creation. Notably, LLMs exhibit remarkable emergence phenomena (Wei et al., 2022b), including zero-shot and in-context learning capabilities (Brown et al., 2020), allowing them to perform tasks without explicit task-specific fine-tuning.

The success of deep learning across vision and language tasks has been largely driven by scaling—deeper networks, larger datasets, and increased computational power—facilitated by open-source frameworks like PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2016), which provide efficient tools for automatic differentiation to streamline gradient computation and accelerate model development (Brown et al., 2020; Kaplan et al., 2020). A key enabler of this scaling is the parallelization of computations on GPUs, which dramatically speeds up matrix operations and enables the efficient training of large models on massive datasets. Modern GPUs, along with distributed training frameworks such as TensorFlow's and PyTorch's distributed module, have made it possible to train models with billions of parameters by distributing computations across multiple devices, reducing training times from weeks to hours. This ability to efficiently scale model depth and size

enables neural networks to learn intricate hierarchical representations, capturing patterns that shallower architectures cannot achieve (He et al., 2016; Mallat, 2016; Dosovitskiy et al., 2021). The increasing depth and parameter scale of modern neural networks has driven remarkable successes in deep learning. However, this depth comes at a cost, introducing significant challenges related to the stability of these models. Key issues include adversarial robustness, training stability, and generalization.

1.2 Seeking stability

Stability is a foundational concept across multiple scientific disciplines, reflecting the ability of a system to maintain consistent behavior under perturbations. The formal study of stability originated in the field of dynamical systems, with Lyapunov's seminal work (Lyapunov, 1892), which introduced methods to assess whether systems return to equilibrium after disturbances. These concepts, initially developed for physical systems, remain highly relevant to modern machine learning, where neural networks can be viewed as discrete dynamical systems during both training and inference.

The work of Tikhonov (1943), who studied the continuity of solutions to ill-posed problems, laying the groundwork for stability analysis in inverse problems. Tikhonov regularization was introduced to stabilize solutions to such problems, ensuring robustness to small perturbations in data. This interplay between stability and robustness has since influenced modern learning theory, where regularization techniques are used to control sensitivity to noise.

Numerical stability was also a concern in computational systems, as highlighted by Von Neumann and Goldstine (1947), who introduced the concept of condition numbers to quantify the sensitivity of solutions to input perturbations. Early foundations of statistical stability can be traced back to Cramér (1946), who formalized how perturbations in data affect estimators. His work introduced key principles such as consistency, ensuring estimators converge to the true parameter as the sample size grows, and asymptotic efficiency, which guarantees minimal variance in large-sample regimes. These properties help mitigate sensitivity to small fluctuations in data, forming a theoretical basis for stable statistical methods. More recent advancements in robust statistics have further refined these stability concepts. In particular, the median-of-means (MoM) estimator (Arkadi S. Nemirovsky, 1983; Mark Jerrum, 1986; Lugosi and Mendelson, 2019) has become a key tool for robust mean estimation under heavy-tailed distributions. Unlike the empirical mean, which can be severely impacted by outliers, MoM partitions data into subsets, computes local

means, and aggregates them using the median. This approach ensures strong concentration properties even in adversarial settings, making it a fundamental technique for stability in modern statistical learning.

Generalization stability refers to a model's ability to remain unaffected by small changes in the training data, such as variations due to sampling, ensuring that the learned function and its performance on unseen data are consistent. Vapnik and Chervonenkis (1971) introduced the concept of the Vapnik-Chervonenkis (VC) dimension, which measures the complexity of a hypothesis class by quantifying the largest number of points it can shatter—that is, classify in all possible ways. Their work provided fundamental theoretical insights into generalization by deriving bounds on the expected generalization error based on the VC dimension. These results established a direct connection between model complexity and generalization, demonstrating that overly complex models tend to overfit while simpler models generalize better. In the late 1970s, Devroye and Wagner (1979a) and Devroye and Wagner (1979b) explored the connection between generalization error and stability, introducing inequalities for leave-one-out and holdout error estimates. Their work implicitly addressed what Kearns and Ron (1999) later termed hypothesis stability, formalizing how the variance of leave-one-out error could be bounded using properties of the learning algorithm. They also demonstrated a connection between finite VC dimension and stability, showing that stable algorithms are inherently generalizable. The PAC framework (Valiant, 1984) provided probabilistic guarantees for generalization, while the PAC-Bayes framework (McAllester, 1999) extended this approach by leveraging prior distributions to derive tighter bounds on the performance of learning algorithms.

Sensitivity analysis (Saltelli et al., 2000) focuses on quantifying the extent to which variations in inputs can influence a system's outputs. It has been widely applied across various disciplines, including statistics, control theory, and mathematical programming where it is referred as perturbation analysis (Bonnans and Shapiro, 1996). The primary motivation for such analyses is to design robust systems that remain resilient to noise or disturbances affecting the inputs. Bousquet and Elisseeff (2002) proposed a sensitivity-based framework to analyze learning algorithms, focusing on how changes in the training set influence the function produced by the algorithm. Unlike classical sensitivity analysis, their framework emphasizes the randomness introduced by sampling and its impact on generalization, bridging stability with probabilistic guarantees. Differential privacy (Dwork et al., 2006) represents a specialized application of sensitivity analysis, where the focus is on bounding the influence of any data point on the output of an algorithm to ensure privacy. By introducing the concept of global sensitivity, differential privacy quantifies the maximum change in the output due to changes in a data point and calibrates

noise to this sensitivity. While its primary goal is privacy protection, the mechanisms of differential privacy inherently limit the dependence of the output on individual inputs, aligning it closely with robustness and stability.

Lipschitz continuity and smoothness emerged as key mathematical tools for analyzing function stability and optimization landscapes. While Lipschitz continuity ensures bounded rates of change in function values, smoothness (characterized by bounded Hessian norm) extends this to gradients (Nesterov, 2003). These properties are fundamental to modern optimization theory, providing convergence guarantees for gradient-based methods and a framework for analyzing learning algorithm stability by ensuring that small input perturbations lead to controlled changes in both function values and their gradients.

1.3 Challenges in deep learning

1.3.1 Stability to sampling noise in data : Generalization

The Lipschitz constant of a neural network plays a crucial role in determining its generalization bounds (Bartlett et al., 2017; Sokolić et al., 2017; Neyshabur et al., 2018). This constant is often bounded by quantities such as the product of the spectral norms of the network's layers, which provides a measure of the network's sensitivity to input perturbations.

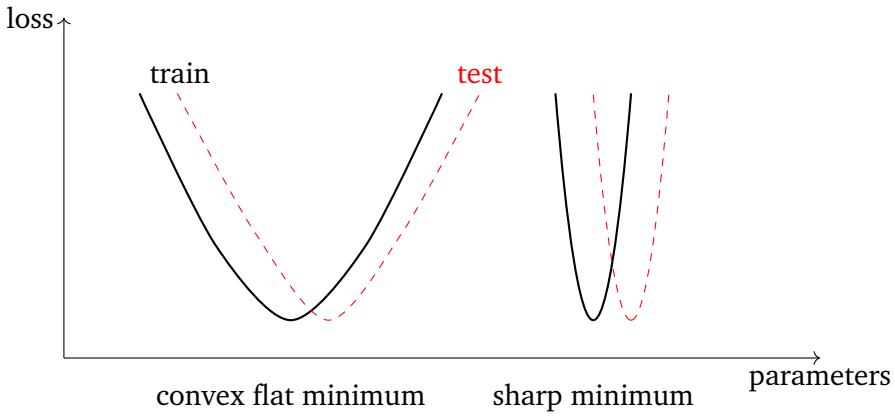


Fig. 1.1.: Depiction of convex flat and sharp minima in one dimension. Both types of minima have the same relative shift between train and test losses.

Beyond Lipschitz continuity, sharpness-based metrics offer additional insights into generalization. Sharpness has long been regarded as a strong indicator of generalization, initially defined as the size of the connected region around the minimum where the training loss remains low (Hochreiter and Schmidhuber, 1997a). Sharpness,

captured by the curvature of the loss landscape's Hessian with respect to parameters, reflects the steepness of the loss landscape. Smaller Hessian values have been empirically associated with improved robustness and generalization (Hochreiter and Schmidhuber, 1997a; Keskar et al., 2017). Flatter minima, characterized by broader and shallower regions in the loss landscape, confer robustness to parameter perturbations and are less sensitive to data distribution shifts. Regularization techniques like Sharpness-Aware Minimization (Foret et al., 2021) explicitly reduce sharpness by minimizing the maximum loss within a local neighborhood of the model's parameters. Other standard methods, such as Weight Decay (WD) (Krogh and Hertz, 1992) have also been shown to encourage flatter minima. Despite these empirical successes, sharpness and flatness metrics are not universally tied to generalization. These measures are sensitive to parameterization and re-scaling effects (Zhang et al., 2017). Furthermore, their relevance heavily depends on both the data and the model (Andriushchenko and Flammarion, 2022). This underscores the need for nuanced approaches when interpreting sharpness-based heuristics. Nevertheless, techniques that reduce sharpness, including SAM and Hessian-based regularization, remain effective practical tools for achieving generalization. Figure 1.1 illustrates the differences between sharp and flat minima, highlighting that flat minima are less sensitive to shifts in the data distribution.

We aim to explore a foundational challenge in neural network design: What principles and regularization techniques can guide the design of neural networks that generalize well to unseen data? While existing methods such as Lipschitz-based regularization and sharpness-aware minimization have shown promise, understanding their interplay and limitations remains an active area of research.

1.3.2 Stability to input: Adversarial robustness

The high capacity and complexity of deep networks make them particularly vulnerable to adversarial examples which are small, carefully crafted perturbations in the input that lead to significant deviations in output (Szegedy et al., 2013; Goodfellow et al., 2015). These perturbations highlight the sensitivity of models to input changes, which becomes increasingly problematic as networks grow larger and more complex. For images, an adversarial attack takes the form of a perturbation imperceptible to the human sight added to the input image. We formulate adversarial robustness as the ability of a model to maintain consistent predictions at inference under input perturbations, ensuring that small changes in the input do not lead to significant changes in the output. Addressing adversarial robustness is critical, particularly in applications where input perturbations can lead to catastrophic failures, see Figure 1.2 for example, where the adversarial perturbation causes a stop sign to be misclassified as a yield sign. The most popular defense to adversarial attack is

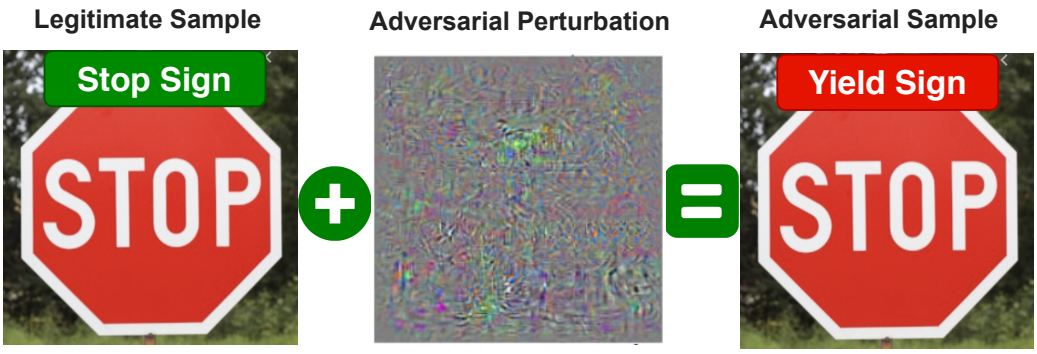


Fig. 1.2.: Example of an adversarial attack on a stop sign. The perturbation is imperceptible to the human eye but causes a misclassification by the network. In this case, the stop sign is misclassified as a yield sign. Figure taken from Ahmad et al. (2022).

adversarial training (Madry et al., 2018) where the network is trained on adversarial examples. This method has been shown to improve robustness to adversarial attacks, but it is computationally expensive and does not provide theoretical guarantees against all types of attacks. Over the past decade, research in adversarial robustness has often resembled a cat-and-mouse game, with increasingly powerful attacks being developed to exploit model vulnerabilities. Notable works in this domain include those by Goodfellow et al. (2015), Kurakin et al. (2016), Carlini and Wagner (2017), and Croce and Hein (2020).

In response to this game, achieving certified adversarial robustness has been a significant focus, with notable contributions such as Raghunathan et al. (2018) and Wong and Kolter (2018). To achieve this, researchers have primarily focused on two categories of methods. The first approach, Randomized Smoothing (RS), comes from differential privacy and is based on randomization (Lecuyer et al., 2018; Li et al., 2018c; Cohen et al., 2019; Pinot et al., 2019; Salman et al., 2019), where noise is injected to the input. This method is the only certified defense on the ImageNet dataset offering non-trivial guarantees, but it suffers from large computational overheads that limit its scalability. The second approach involves controlling the relationship between the Lipschitz constant of the network and the prediction margins (Tsuzuku et al., 2018), following this work involves constructing 1-Lipschitz layers using specific linear transformations (Cisse et al., 2017; Anil et al., 2019; Li et al., 2019; Li et al., 2020; Singla and Feizi, 2021b; Singla and Feizi, 2021c; Trockman and Kolter, 2021) relying on that the Lipschitz constant of a function quantifies the maximum sensitivity of the function's output to changes in its input. By bounding the Lipschitz constant of neural networks, one can compute a certification radius around any given point by establishing maximum sensitivity around the input for a given perturbation's budget. While these methods offer a more direct and

theoretically grounded approach, they are hindered by limited performance and scalability.

Given these limitations, a critical question arises: How can we improve existing methods for certified adversarial robustness to address their computational and scalability challenges while maintaining strong guarantees?

1.3.3 Training stability

Stability during training can be defined as low sensitivity to perturbations in the input, parameters, or gradients noise, ensuring that small changes do not lead to drastic variations in the training's output. Training stability is crucial for achieving consistent performance across different runs and datasets, as well as for enabling efficient optimization and convergence to good solutions.

The increasing scale of deep networks, in terms of depth and parameter count, often exacerbates training instability. Training deep networks is inherently challenging due to issues such as exploding and vanishing gradients (Pascanu et al., 2013), as well as sensitivity to initialization (Glorot and Bengio, 2010). These challenges become especially pronounced in large-scale architectures, where instability can hinder convergence and lead to suboptimal solutions. One of the earliest breakthroughs in addressing these issues was the development of improved activation functions. Early architectures relied on sigmoid and tanh activations, which often suffered from saturation issues, resulting in vanishing gradients (Hochreiter and Schmidhuber, 1997b). The introduction of ReLU (Rectified Linear Unit) (Nair and Hinton, 2010) marked a turning point by allowing gradients to flow more effectively. Subsequent variants, such as Leaky ReLU (Maas et al., 2013), Parametric ReLU (He et al., 2015), and GELU (Gaussian Error Linear Unit) (Hendrycks and Gimpel, 2016), further improved gradient flow and facilitated the training of deeper networks. Building on these advances, residual connections, introduced with ResNet (He et al., 2016), offered another transformative solution to training instability. While normalization layers such as batch normalization (Ioffe and Szegedy, 2015), layer normalization (Ba et al., 2016), and group normalization (Wu and He, 2018) help stabilize activations and address the vanishing gradient problem, residual connections tackle a related but distinct challenge. By enabling gradients and signals to bypass one or more layers, residual connections mitigate the degradation of the noise-to-signal ratio as information propagates through deep networks (Balduzzi et al., 2017). This mechanism preserves critical information during both forward and backward passes, helping stabilize training in very deep architectures and enabling faster convergence with higher performance.

In addition to architectural innovations, advancements in optimization techniques have been pivotal in improving training stability. Optimizers such as Adam (Kingma and Ba, 2015) and its variants (e.g., AdamW (Loshchilov and Hutter, 2019) and LAMB (You et al., 2020)) have become the standard for training deep models, providing faster convergence and better handling of large parameter spaces. Techniques like learning rate schedules, momentum (Polyak, 1964), and adaptive gradient methods (e.g., Adagrad (Duchi et al., 2011) and RMSProp (Tieleman and Hinton, 2012)) further enhance stability and efficiency. Additionally, normalization layers like batch normalization (Ioffe and Szegedy, 2015) not only stabilize activations but also enable the use of larger learning rates, further boosting training dynamics in deep networks.

While these techniques improve training stability, they often rely on complex heuristics and tuning. This underscores the need for architectures that are inherently stable by design, reducing reliance on external tricks. Lipschitz regularization or normalization techniques, such as spectral normalization (Miyato et al., 2018), have been widely used to stabilize GAN training by constraining the gradient norms, preventing exploding gradients, and ensuring smoother updates for the generator and discriminator.

Lipschitz networks provide robustness to input perturbations and ensure stable gradient flow that does not explode, making them strong candidates for improving stability by design. This naturally raises the question: How can we build efficient Lipschitz neural networks?

1.4 Contributions and outline

This thesis adopts sensitivity analysis as both a guiding framework and a practical tool to address stability challenges and answer the questions raised in the previous section. By leveraging derivative-based methods, such as Lipschitz constant estimation and Hessian analysis, alongside variance-based techniques like noise injection, we quantify and characterize sensitivity at various levels of deep models, proposing actionable solutions through architecture design that inherently minimizes sensitivity.

Specifically, this work develops Lipschitz networks to control output sensitivity to input perturbations, offering solutions to challenges in generalization, adversarial robustness, and training stability. Furthermore, we introduce Hessian-bounded loss regularization to smooth the loss landscape, reducing parameter sensitivity and improving optimization dynamics. Randomized smoothing is employed to

enhance robustness at decision boundaries. Finally, we explore the theoretical and practical connections between these approaches and the Lipschitz continuity of neural networks, unifying them under a common framework.

We aim to address those challenges in the following chapters:

- **Chapter 2** This chapter provides an overview of Lipschitz networks, certified robustness in the context of Lipschitz networks, randomized smoothing, and regularization techniques aimed at improving generalization.

- **Chapter 3** This chapter focuses on spectral norm computation, a key step in constructing Lipschitz networks. We introduce *Gram iteration*, an efficient algorithm for estimating the spectral norm of dense and convolutional layers under different padding schemes. Additionally, we establish theoretical connections between the spectral norms of circular and zero-padding convolutions. These results lay the foundation for Lipschitz-constrained architectures.

- **Chapter 4** In this chapter, we explore novel approaches for computing and regularizing the Lipschitz constant of convolutional layers, including the development of novel by design Lipschitz layers, spectrally rescaled, based on *Gram iteration*. In the same line of work of Salman et al., 2019, we view the randomized smoothing procedure as a way to obtain Lipschitz networks through Weierstrass transform and explore its connections with Lipschitz continuity.

- **Chapter 5** In this chapter, we propose methods to certify robustness by designing Lipschitz networks using spectrally rescaled layers. Furthermore, we demonstrate the effective integration of randomized smoothing with Lipschitz networks, leveraging their inherent stability to provide better probabilistic guarantees. Additionally, we extend the simplex mapping in randomized smoothing to a scaled simplex mapping, enabling improved margin-to-Lipschitz ratio and robustness certification for a broader range of perturbations.

- **Chapter 6** Randomized smoothing introduces additional stochasticity and variance, necessitating careful management to ensure robust performance. This chapter focuses on risk management in randomized smoothing, with an emphasis on statistical certification procedures and computational efficiency. We tackle the Lipschitz-variance-margin tradeoff for Randomized Smoothing (LVM-RS), demonstrating how a balanced approach can achieve both low variance and high margins for robust models. To this end, we propose new techniques such as LVM-RS and class partitioning methods (CPM) for

confidence intervals, which enhance robustness while significantly reducing computational overhead.

- **Chapter 7** In this chapter, Lipschitz regularization is used to enhance generalization. We then introduce Hessian curvature regularization as a means to promote flat minima to improve generalization, leveraging the Weierstrass transform applied to the gradient of the loss with respect to parameters. Furthermore, we establish a connection between Lipschitz networks and flat minima in their loss landscape, showing how these architectures naturally promote broader, flatter solutions. Finally, we propose the Activation Decay method as a deterministic approach to loss smoothing, complementing Lipschitz regularization.

The work presented in this thesis is based on results published in the following peer-reviewed conferences, as well as preprints under review:

Contributions included in the manuscript:

- *In Chapter 3 and Chapter 7:*
  B.Delattre, Q.Barthélemy, A.Araujo, and A.Allauzen, "Efficient Bound of Lipschitz Constant for Convolutional Layers by Gram Iteration," Proceedings of the 40th International Conference on Machine Learning (ICML), 2023.
- *In Chapter 3 and Chapter 4:*
  B.Delattre, Q. Barthélemy, and A.Allauzen, "Spectral Norm of Convolutional Layers with Circular and Zero Paddings," (under review) arXiv preprint, 2024.
- *In Chapter 4, Chapter 6, and Chapter 5:*
  B.Delattre, A.Araujo, Q.Barthélemy, and A.Allauzen, "The Lipschitz-Variance-Margin Tradeoff for Enhanced Randomized Smoothing," International Conference on Learning Representations (ICLR), 2024.
  B.Delattre, P.Caillon, Q.Barthélemy, E.Fagnou, and A.Allauzen, "Bridging the Theoretical Gap in Randomized Smoothing," Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS), 2025.
- *In Chapter 7:*
  B.Delattre, Q.Barthélemy, P.Caillon, E.Fagnou, and A.Allauzen, "Regularization by Activation Decay to Enhance Generalization," (under review), 2025.

1.5 Additional contributions

Building on Lyapunov's stability theory, Haber and Ruthotto (2018) interpreted deep learning as a parameter estimation problem in nonlinear dynamical systems. They

proposed architectures inspired by ordinary differential equations (ODEs) to address numerical instabilities, such as exploding and vanishing gradients and designed well-posed architectures for arbitrarily deep networks.

Inspired by this perspective, we developed a unified framework for constructing 1-Lipschitz Neural Networks by interpreting Residual Networks as continuous-time dynamical systems. This framework reveals that many existing methods are specific instances of this approach. Moreover, we showed that ResNet flows derived from convex potentials naturally define 1-Lipschitz transformations, leading to the introduction of the *Convex Potential Layer (CPL)*. This method enables ℓ_2-provable defenses against adversarial examples and allows for the stable training of extremely deep ResNets, with over 1000 layers, without relying on techniques such as batch normalization or gradient clipping.

We further expanded this work by unifying recent advancements in 1-Lipschitz neural networks, including methods based on convex potentials and other concurrent approaches, under a novel algebraic framework. By leveraging a common semidefinite programming (SDP) condition, we demonstrated that existing techniques can be derived as special cases and introduced new parameterizations for Lipschitz layers using the Gershgorin circle theorem. This approach, termed *SDP-based Lipschitz Layers (SLL)*, generalizes convex potential layers.

These additional works have been disseminated in peer-reviewed venues:

Contributions not included in the manuscript:

- *L.Meunier*, B.Delattre*, A.Araujo*, and A.Allauzen,*
  "A Dynamical System Perspective for Lipschitz Neural Networks,"
  Proceedings of the 39th International Conference on Machine Learning (ICML), 2022.
- *A.Araujo*, A.J.Havens*, B.Delattre, A.Allauzen, and B.Hu,*
  "A Unified Algebraic Perspective on Lipschitz Neural Networks,"
  International Conference on Learning Representations (ICLR), 2023.

2 Background

Contents

We define a feed-forward neural network f with L layers by

$$f = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}, \tag{2.1}$$

where each layer consists of a weight matrix $\mathbf{W}^{(l)}$ followed by a nonlinearity $\rho^{(l)}$, typically ReLU or GELU,

$$f^{(l)}(\mathbf{z}) = \begin{cases} \rho^{(l)}\big(\mathbf{W}^{(l)}\,\mathbf{z}\big), & l = 1, \dots, L-1, \\ \mathbf{W}^{(L)}\,\mathbf{z}, & l = L. \end{cases}$$

We denote the spectral norm of $\mathbf{W}^{(l)}$ by $\|\mathbf{W}^{(l)}\|_2$. For simplicity, we omit biases, which are implicitly accounted for. We note θ as the set of all parameters in the network, $\theta = \{\mathbf{W}^{(l)}\}_{l=1}^{L}$. For an input $\mathbf{x}$, define intermediate outputs inductively:

$$h^{(0)}(\mathbf{x}) = \mathbf{x}, \quad h^{(l)}(\mathbf{x}) = f^{(l)}\big(h^{(l-1)}(\mathbf{x})\big), \quad l = 1, \dots, L.$$

Then $f(\mathbf{x}) = h^{(L)}(\mathbf{x})$.

A key property of neural networks is their Lipschitz constant, which quantifies how much the output of the network can change with respect to small changes in input. Formally, the Lipschitz constant of f is defined as:

$$\mathrm{L}(f) = \sup_{\mathbf{x} \neq \mathbf{x}'} \frac{\|f(\mathbf{x}) - f(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \ ,$$

and provides an upper bound on how much the output can change relative to variations in input. A high Lipschitz constant implies large output variations under small input changes, making networks more susceptible to adversarial attacks (Szegedy et al., 2013; Goodfellow et al., 2015) and training instabilities. This property is directly linked to the maximum gradient norm of the network, as for differentiable functions: $\mathrm{L}(f) \geq \sup_{\mathbf{x}} \|\nabla f(\mathbf{x})\|$. Excessive gradient magnitudes w.r.t parameters θ can lead to unstable optimization, known as the exploding gradient problem (Pascanu et al., 2013). Regularizing the Lipschitz constant mitigates such issues by constraining gradient growth, promoting smoother and more stable training. Beyond stability, Lipschitz continuity plays a crucial role in generalization by limiting function complexity and reducing overfitting (Bartlett et al., 2017). It also provides theoretical guarantees on model behavior under input perturbations (Virmaux and Scaman, 2018), with applications in adversarial robustness (Tsuzuku et al., 2018), generalization bounds (Bartlett et al., 2017), and optimization (Fazlyab et al., 2019).

In this chapter, we begin by presenting techniques for estimating the Lipschitz constant of neural networks, analyzing their strengths and limitations. This is followed by methods for enforcing Lipschitz continuity through regularization during training, indirectly constraining model behavior. Additionally, we highlight architectural strategies that ensure Lipschitz continuity by design, removing the need for post-training adjustments. The convolution with Gaussian kernel also known as Weierstrass transform is also explored as a smoothing mechanism to control the Lipschitz constant of the network.

Next, we examine certified robustness, which guarantees model stability under adversarial perturbations. We review how Lipschitz networks achieve deterministic certification by leveraging their bounded Lipschitz constant. In parallel, we discuss randomized smoothing, a probabilistic method for robustness certification, showing how smoothing techniques complement Lipschitz constraints to improve resilience.

The final section of this chapter explores regularization techniques for generalization which aim to control model complexity. Lipschitz regularization constrains sensitivity to input perturbations through spectral norm penalties, projection methods, and architectural constraints. Flat minima are also closely linked to better generalization, with techniques like sharpness-aware minimization promoting smoother loss landscapes by reducing sharpness. Finally, loss smoothing and noise-based regular-

ization introduce stochasticity, leveraging techniques such as dropout and perturbed gradient descent.

2.1 Lipschitz constraint

Lipschitz constraints can be expressed under various norms, each inducing different geometric and robustness properties. In this work, we adopt the conventional but ultimately arbitrary choice of the ℓ_2-norm. This selection is primarily motivated by its correspondence with the spectral norm for linear transformations, which provides a tractable and principled proxy for controlling layerwise sensitivity. Moreover, Gaussian noise—ubiquitous in both training procedures and certified defenses—spreads isotropically, naturally aligning with the ℓ_2 metric. These factors make the ℓ_2-norm a practical and theoretically grounded foundation for studying Lipschitz regularization. However, this choice is not universal. In adversarial robustness, the ℓ_∞-norm captures worst-case, high-frequency perturbations (Wong et al., 2018), and several works have developed Lipschitz architectures explicitly tailored to this norm to obtain stronger certified guarantees (Zhang et al., 2021a; Zhang et al., 2022a; Zhang et al., 2022b). More broadly, Gouk et al. (2021) show that alternative norms such as ℓ_1 and ℓ_∞ can be effective for promoting robustness and stability, depending on the inductive bias or application domain. The choice of norm should thus reflect the structure of the task and the form of perturbations one aims to guard against.

Formally, the Lipschitz constant of a function f with respect to the ℓ_2-norm is denoted as

$$\mathrm{L}(f) = \sup_{\mathbf{x}\neq\mathbf{y}} \frac{\|f(\mathbf{x}) - f(\mathbf{y})\|_2}{\|\mathbf{x} - \mathbf{y}\|_2} \ .$$

For a linear transformation with weight matrix $\mathbf{W}$, this constant coincides with the *spectral norm* of $\mathbf{W}$, defined as

$$\|\mathbf{W}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{W}\mathbf{x}\|_2 \ ,$$

which is equal to the largest singular value of $\mathbf{W}$.

The spectral norm is closely related to the *spectral radius* of a matrix, defined as

$$\rho(\mathbf{W}) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } \mathbf{W}\} \ .$$

In general, we have $\rho(\mathbf{W}) \leq \|\mathbf{W}\|_2$, with equality when $\mathbf{W}$ is normal (i.e., $\mathbf{W}^\top\mathbf{W} = \mathbf{W}\mathbf{W}^\top$). While the spectral norm governs the worst-case amplification of vector norms, the spectral radius measures the asymptotic growth rate of powers of $\mathbf{W}$, a property relevant for stability analysis in iterative processes.

In Chapter 3, we present existing methods and introduce *Gram iteration*, a novel method for estimating spectral norms that improves upon existing techniques. Most non-linear activation functions, such as ReLU, sigmoid, or GELU, are 1-Lipschitz.

Lipschitz networks require increased capacity: Scaling the number of parameters is essential for Lipschitz networks to balance smoothness constraints with expressivity. By increasing model width or depth, one can compensate for the representational limitations introduced by enforcing strict Lipschitz bounds, thus preserving the network's ability to fit complex functions while maintaining robustness and stability guarantees. While Lipschitz continuity provides strong theoretical guarantees for robustness and generalization, it limits the function space, requiring significantly larger models for competitive performance. Bubeck et al. (2021) and Bubeck and Sellke (2021) showed that smooth decision boundaries under Lipschitz constraints require d-times more parameters than non-smooth ones, where d is the data dimension. Thus, achieving robustness and certification necessitates scaling model capacity to maintain expressivity. Current Lipschitz-based certification methods often fail to certify points that are in fact robust—i.e., inputs for which the model's prediction remains constant under all admissible perturbations within a norm ball (Yang et al., 2020b). Addressing these challenges requires scaling Lipschitz networks to fully leverage their theoretical advantages. Béthune (2024) also explore the expressivity of Lipschitz networks, focusing on orthogonal architectures. They highlight that these architectures require more parameters than standard ones to achieve equivalent expressivity.

Lipschitz beyond adversarial robustness and generalization While its primary utility lies in generalization and adversarial robustness, robustness to perturbations is also of great use in several applications such as reinforcement learning (Brunke et al., 2022), control algorithms (Shi et al., 2018), or other applications that leverage Lipschitz continuity, such as one-class classification with signed distance functions (Béthune et al., 2023). For instance, Wasserstein Generative Adversarial Networks (WGANs) (Arjovsky et al., 2017) leverage the Wasserstein-1 distance to improve GAN training stability, which requires the discriminator to be 1-Lipschitz. Enforcing this constraint, critical for valid distance computation, is typically achieved through techniques such as weight clipping (Arjovsky et al., 2017) or gradient penalty (Gulrajani et al., 2017). Beyond GANs, Lipschitz constraints have been used in tasks requiring differential privacy. For example, (Béthune et al., 2024) propose Lipschitz-SGD to enforce differential privacy by bounding gradient sensitivity, while Ghazanfari et al. (2024) introduces a provably Lipschitz continuous image similarity measure to enhance the robustness of DreamSim (Fu et al., 2023) under adversarial attacks. Also, Lipschitz networks are used as a backbone architecture for normal-

izing flow models (Verine et al., 2023), where the Lipschitz constraint ensures the invertibility and stability of the flow.

2.1.1 Estimating the Lipschitz constant

Computing the exact Lipschitz constant of a neural network is NP-hard (Virmaux and Scaman, 2018) (even for a 2-layer multi-layer perceptron), primarily due to the exponential proliferation of piecewise linear regions created by activations like ReLU. Each ReLU introduces new subregions in the input space, and finding the global maximum over all such regions quickly becomes intractable as networks grow in size. While some methods exist to estimate or to bound it (Fazlyab et al., 2019; Latorre et al., 2020), their computational cost makes them impractical for deep architectures. The complexity of modern networks, with their exponentially growing number of parameters and layers, further exacerbates this challenge. Approximation techniques, such as bound propagation (Zhang et al., 2019a; Jordan and Dimakis, 2020; Shi et al., 2022), have been proposed to address scalability. However, these methods often provide loose or overly conservative bounds, limiting their utility during training. Thus, efficiently estimating and controlling Lipschitz constants in deep learning remains an open and active area of research.

Measuring Local Lipschitzness Local Lipschitzness quantifies how much a function's output can change within a small neighborhood of the input. In contrast to the global Lipschitz constant—which provides a uniform upper bound over the entire domain and can be overly conservative—local Lipschitz constants offer a more refined measure of sensitivity in restricted regions.

Definition 2.1.1 (Local Lipschitz constant)**.**
Let $f : \mathbb{R}^d \to \mathbb{R}^m$ be a function and let $\|\cdot\|$ be a norm on $\mathbb{R}^d$. Given an open set $\mathcal{B} \subset \mathbb{R}^d$, the local Lipschitz constant of f over $\mathcal{B}$ is defined as

$$\mathrm{L}(f, \mathcal{B}) = \sup_{\substack{\mathbf{x}, \mathbf{x}' \in \mathcal{B} \\ \mathbf{x} \neq \mathbf{x}'}} \frac{\|f(\mathbf{x}) - f(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \quad . \tag{2.2}$$

We say that f is locally Lipschitz over $\mathcal{B}$ if $\mathrm{L}(f, \mathcal{B}) < \infty$. When $\mathcal{B} = \mathbb{R}^d$, we write $\mathrm{L}(f) \coloneqq \mathrm{L}(f, \mathbb{R}^d)$ and refer to it as the global Lipschitz constant.

Remark The restriction $\mathbf{x} \neq \mathbf{x}'$ ensures that the denominator is non-zero. In the limit $\mathbf{x}' \to \mathbf{x}$, the Lipschitz ratio approaches the norm of the Jacobian when f is differentiable. In particular, one has

$$\mathrm{L}(f, \mathcal{B}) \geq \sup_{\mathbf{x} \in \mathcal{B}} \|J_f(\mathbf{x})\|,$$

where $\|J_f(\mathbf{x})\|$ denotes the operator norm of the Jacobian at $\mathbf{x}$.

To evaluate the local sensitivity of a neural network f, one can estimate its local Lipschitz constant over a dataset. Let $\mathcal{D}$ be the data distribution, and let $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ denote an empirical dataset drawn i.i.d. from $\mathcal{D}$. Following Yang et al. (2020b), the local Lipschitz constant at a point $\mathbf{x} \sim \mathcal{D}$, relative to a perturbation set $\mathcal{B}(\mathbf{x}) \subset \mathbb{R}^d$, is defined in expectation as:

$$\mathrm{L}(f, \mathcal{B}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\sup_{\mathbf{x}' \in \mathcal{B}(\mathbf{x})} \frac{\|f(\mathbf{x}) - f(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \right],$$

where $\mathcal{B}(\mathbf{x})$ typically denotes a norm-bounded neighborhood around $\mathbf{x}$, such as $\mathcal{B}(\mathbf{x}) = \{\mathbf{x} + \eta \in \mathbb{R}^d \mid \|\eta\|_2 \leq \epsilon\}$. In practice, this expectation is approximated by an empirical mean over the dataset $\mathcal{S}$, yielding the empirical Lipschitz estimate:

$$\mathrm{L}_{\mathrm{emp}}(f, \mathcal{B}) = \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{x}_i' \in \mathcal{B}(\mathbf{x}_i)} \frac{\|f(\mathbf{x}_i) - f(\mathbf{x}_i')\|}{\|\mathbf{x}_i - \mathbf{x}_i'\|}. \tag{2.3}$$

This quantity serves as a tractable surrogate for measuring the local smoothness of f around points in the dataset. To approximate the inner maximization in (2.3), Yang et al. (2020b) employ a projected gradient ascent procedure similar to PGD attacks (see Algorithm 1). While effective in practice, this approach does not yield a guaranteed upper bound on the local Lipschitz constant, but only an empirical lower bound based on approximate maximizers.

A product upper bound for Lipschitz constant The Lipschitz constant of a neural network arises from the composition of its layers (Equation 2.1), the composition property of Lipschitz functions offers a starting point: the product of individual layer Lipschitz constants provides an upper bound for the global Lipschitz constant.

Definition 2.1.2 (Product Upper Bound (PUB) for Lipschitz constant)**.**
A simple upper bound for the Lipschitz constant of a neural network is given by the product of individual layer Lipschitz constants:

$$\mathrm{L}(f) \leq \mathrm{PUB}(f) = \prod_{l=1}^{L} \mathrm{L}(f^{(l)}) \,. \tag{2.4}$$

Here, $\mathrm{L}(f)$ *is the Lipschitz constant of the entire network, and* $\mathrm{L}(f^{(l)})$ *is the Lipschitz constant of the* l*-th layer.*

For ReLU networks, the product of layer-wise Lipschitz constants generally overestimates the true Lipschitz constant. While linear layers can in principle align to amplify the same direction, ReLU layers often zero out parts of the input, disrupting this alignment and preventing each layer's maximal scaling factor from fully accumulating. Even for linear networks, the product bound holds strictly only under the perfect alignment of transformations—for instance, in homothetic (scaling) layers or orthogonal transformations which is rarely the case with standard networks.
The product upper bound was first used in (Szegedy et al., 2013) and has since been applied in generalization bounds (Bartlett et al., 2017) and the design of Lipschitz networks (Tsuzuku et al., 2018; Trockman and Kolter, 2021; Meunier et al., 2022; Araujo et al., 2023). While this bound is often loose, it serves as the foundation for designing scalable techniques to estimate and regularize Lipschitz constants, with applications in stability, robustness, and generalization. Using this bound, one can leverage the spectral norm of each layer to control the network's Lipschitz constant or design Lipschitz layers directly. The PUB serves as a starting point for the design and regularization of Lipschitz layers.

Lipschitz constant of attention The PUB can be used when the Lipschitz constant of each layer is tractable to compute, such as for linear layers or layers with known Lipschitz constants. However, for self-attention blocks, which form the core of transformer architectures (Vaswani et al., 2017), deriving Lipschitz properties is less straightforward. The self-attention score is given by

$$\mathbf{X} \mapsto= \mathrm{softmax}\left(\frac{\mathbf{W}_K\mathbf{X}(\mathbf{W}_Q\mathbf{X})^\top}{\sqrt{d}}\right),$$

where the input vector $\mathbf{x}$ is reshaped into a matrix $\mathbf{X}$, and $\mathbf{W}_K, \mathbf{W}_Q$ are respectively the key and query weight matrices. While the $\mathrm{softmax}$ function itself is 1-Lipschitz (Gao and Pavel, 2018), the quadratic form $\mathbf{W}_K\mathbf{X}(\mathbf{W}_Q\mathbf{X})^\top$ introduces a bilinear dependency on $\mathbf{X}$, leading to potential unbounded growth in the operator norm. This makes it difficult to control the Lipschitz constant of the self-attention layer, as the spectral norm of $\mathbf{X} \mapsto \mathbf{W}_K\mathbf{X}(\mathbf{W}_Q\mathbf{X})^\top$ can scale with $\|\mathbf{X}\|^2$. Unlike linear layers, where the Lipschitz bound is determined by the spectral norm of a single weight matrix, the interaction between $\mathbf{W}_K, \mathbf{W}_Q$ and the input $\mathbf{X}$ complicates the analysis. The work of Kim et al. (2021) explores alternative Lipschitz-constrained formulations of the attention block, while several studies investigate the Lipschitz properties of self-attention for bounded input domains (Havens et al., 2024) or with respect to Wasserstein distance (Vuckovic et al., 2021; Castin et al., 2024). However,

deriving Lipschitz bounds for attention mechanisms remains an open challenge, with implications for the robustness and stability of transformer models.

Lipschitz Constant of Normalization Layers. Normalization layers such as BatchNorm (Ioffe and Szegedy, 2015), LayerNorm (Ba et al., 2016), and GroupNorm (Wu and He, 2018) are widely used in deep learning to stabilize training and improve convergence. For BatchNorm, each activation is normalized as

$$\mathrm{BN}(\mathbf{x}) = \gamma \frac{\mathbf{x} - \mu_j}{\sigma_j} + \beta,$$

where μ_j and σ_j are the empirical mean and standard deviation across the batch for feature dimension j, and γ, β are learnable affine parameters. Santurkar et al. (2018) showed that BatchNorm alters the geometry of the loss by modifying gradient norms. For a loss $\mathcal{L}$ and pre-activation $\mathbf{y}_i$ of the i-th input in a mini-batch of size m, the gradient norm of the transformed loss $\hat{\mathcal{L}}$ satisfies

$$\left\|\nabla_{\mathbf{y}_i}\hat{\mathcal{L}}\right\|^2 \leq \frac{\gamma^2}{\sigma_j^2}\left(\|\nabla_{\mathbf{y}_i}\mathcal{L}\|^2 - \frac{1}{m}\langle \mathbf{1}, \nabla_{\mathbf{y}_i}\mathcal{L}\rangle^2 - \frac{1}{m}\langle \nabla_{\mathbf{y}_i}\mathcal{L}, \hat{\mathbf{y}}_j\rangle^2\right),$$

where $\hat{\mathbf{y}}_j$ denotes the normalized activation direction. This reveals that BatchNorm rescales gradients by γ/σ_j while suppressing components aligned with the constant vector and the activation axis, effectively reducing the local Lipschitz constant and stabilizing training. Second-order analysis confirms that the Hessian norm is similarly attenuated, bounded by γ^2/σ_j^2.

However, despite these benefits, BatchNorm does not ensure Lipschitz continuity during training. Because μ_j and σ_j are computed from the current mini-batch, the transformation becomes a stochastic function that depends on the entire batch. If the variance σ_j^2 is small, the scaling factor γ/σ_j may become arbitrarily large, leading to unbounded local Lipschitz constants. Moreover, the interaction between examples introduces coupling effects that complicate theoretical analysis. The discrepancy between training (with batch statistics) and inference (with frozen statistics) further breaks Lipschitz consistency across phases. Similar caveats apply to other normalization schemes like LayerNorm or GroupNorm, although they normalize across different axes. In summary, while normalization layers improve gradient flow and training stability, they complicate explicit Lipschitz control and may be incompatible with Lipschitz-regularized objectives.

In the following sections, we present regularization methods for enforcing Lipschitz constraints on neural networks.

2.1.2 Lipschitz through regularization

Gradient Norm Penalty Gradient norm penalties (Drucker and LeCun, 1992; Gulrajani et al., 2017; Ross and Doshi-Velez, 2018) offer an indirect way to control the Lipschitz constant by regularizing the norm of the input gradients during training. This approach has been successfully applied in adversarial robustness and Wasserstein GANs. A typical regularization term, introduced by Gulrajani et al. (2017), penalizes deviations of the gradient norm from unity:

$$\mathcal{L}_{\text{GP}} = \frac{1}{m}\sum_{i=1}^{m}\left(\|\nabla_{\mathbf{x}} f(\mathbf{x}_i)\|_2 - 1\right)^2 ,$$

where the expectation is approximated by the empirical average over a batch of m input points. This regularizer encourages the function to be approximately 1-Lipschitz in regions traversed during training, without explicitly constraining the spectral norm of the weights. However it is computationally expensive because it requires a double backpropagation which involves a large computational graph.
To circumvent this cost regularization on individual layers has been proposed by Yoshida and Miyato, 2017 by proposing spectral normalization which directly constrains the Lipschitz constant of the layer by bounding the spectral norm of the weight matrix.

$$\mathcal{L}_{\text{SN}} = \sum_{i=1}^{L}\sigma_1(\mathbf{W}^{(l)})$$

Other techniques, such as soft regularization promote orthogonality (Huang et al., 2020; Wang et al., 2020), with regularization loss like:

$$\mathcal{L}_{\text{ortho}} = \sum_{l=1}^{L}\|\mathbf{W}^{(l)}\mathbf{W}^{(l)\top} - \mathbf{I}\|_F^2 .$$

The work of Arjovsky et al. (2017) uses weight clipping to ensure the Lipschitz constraint of the discriminator in Wasserstein GANs. This method is simple and effective, but it can lead to optimization issues and model instability. Other approaches rely on projection P after weights updates but do not alter gradient computation, for instance, Salimans and Kingma (2016) used weight normalization to constrain the Lipschitz constant of the network,

$$\mathrm{P}(\mathbf{W}^{(l)}) = \frac{\mathbf{W}^{(l)}}{\|\mathbf{W}^{(l)}\|_{\text{F}}} .$$

But Frobenius norm is a loose bound on the spectral norm which is the Lipschitz constant of the layer. Miyato et al. (2018) refines this method by using spectral normalization to enforce Lipschitz constraints on the discriminator of GANs,

$$\mathrm{P}(\mathbf{W}^{(l)}) = \frac{\mathbf{W}^{(l)}}{\sigma_1(\mathbf{W}^{(l)})} \ .$$

In the same flavor Gouk et al. (2021) proposed a projection-based method to enforce Lipschitz constraints on individual layers. enforce the Lipschitz constraint on a layer, Gouk et al. (2021) applies a projection P onto the set of matrices with a Lipschitz constant less than or equal to λ. This projection is applied to the weight matrices $\mathbf{W}^{(l)}$ after each training update:

$$\mathrm{P}(\mathbf{W}^{(l)}) = \frac{\mathbf{W}^{(l)}}{\max\left(1, \frac{\|\mathbf{W}^{(l)}\|_p}{\lambda}\right)}$$

With λ a hyperparameter controlling the layer Lipschitz constant, and for $p = 1, 2, \infty$.

Lipschitz constant approximation methods (Fazlyab et al., 2019) provide tighter bounds by incorporating upper bounds of the constants into the loss function using semidefinite programming, though their high computational cost makes them less practical for large-scale models. Those normalization methods are efficient from a computational standpoint, but they can be too restrictive, and lead to vanishing gradients, the maximum direction is controlled but the average direction can collapse to zero.

Margin-based regularization (Tsuzuku et al., 2018) has been explored to improve robustness by maximizing the margin between decision boundaries. The work of Leino and Fredrikson (2021) and Leino et al. (2021) extends this paradigm by introducing Lipschitz bound (PUB) regularization directly integrated into the loss function. Those approaches conjugate regularization training with by design Lipschitz layers to ensure robustness and training stability.

2.1.3 Lipschitz by design

By design Lipschitz layers allow for certified robustness and stability without the need for post-hoc adjustments or regularization during training. These layers are designed to be 1-Lipschitz, ensuring that the network as a whole is also 1-Lipschitz leveraging the PUB:

$$\mathrm{L}(f) \leq \prod_{i=1}^{L} \mathrm{L}(f^{(l)}) = 1 \ .$$

The main difficulty with Lipschitz layers lies in deriving such layers for complex architectures like convolutional layers, residual layers, and attention mechanisms. Dense layers are more straightforward to derive and analyze in terms of their Lipschitz constants as we can study directly the matrix representing the transformation. First, we consider a linear layer,

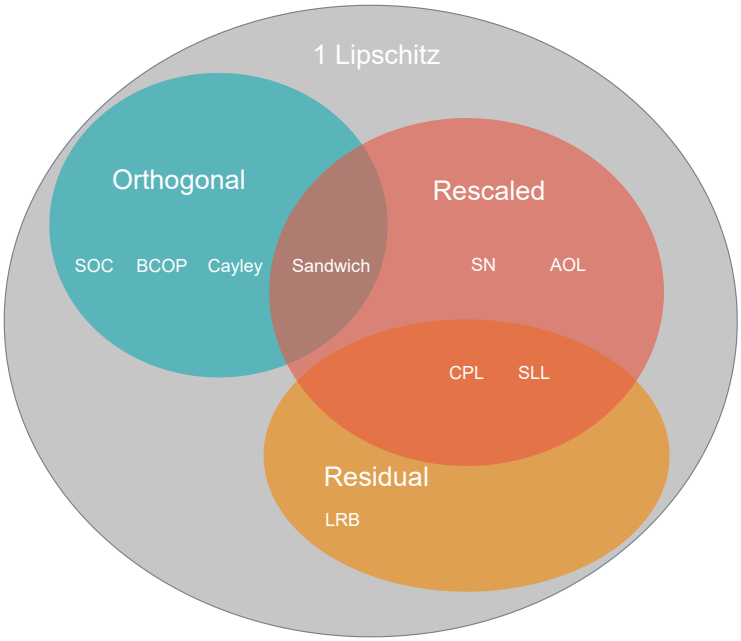


Fig. 2.1.: Venn diagram illustrating the categorization of Lipschitz layers, including scaled layers, orthogonal layers, and residual layers.

$$\mathbf{x} \mapsto \mathbf{W}\mathbf{x}$$

where $\mathbf{W}$ is the weight matrix and input $\mathbf{x} \in \mathbb{R}^d$. Here $\mathbf{W}$ can be a dense matrix or a sparse matrix such as for the convolutional layer. Here the bias is implicitly included.

Orthogonal layers One convenient way to ensure the Lipschitz constant is to enforce orthogonality on the weight matrix, it is a good property for training properties as the layer is norm preserving. This can be done by optimizing the weight matrix onto the orthogonal manifold (Anil et al., 2019; Ablin and Peyré, 2022). Another method is to rely on orthogonal layers by design, these methods aim to project weight matrices onto an orthogonal space. A key challenge arises with convolutional layers, where the operation cannot be directly expressed as matrix multiplication, because the matrix representing the convolutional operation is usually too large to be stored in memory: the size of the matrix grows with $O(n^4)$ where n is the input dimension ($\mathbf{x} \in \mathbb{R}$). For images from CIFAR-10 or ImageNet $n = 3 \times 32 \times 32$ and respectively $3 \times 224 \times 224$, which is infeasible to store in memory.

To address this, Reshaped Kernel Orthogonalization (RKO) methods (Li et al., 2019) enforce orthogonality on the reshaped convolutional filter K. However, making K orthogonal does not guarantee that the convolutional layer itself is orthogonal, as the convolution operation introduces additional dependencies. Instead, the Lipschitz constant of the convolutional layer is upper bounded by the Lipschitz constant of the reshaped kernel, up to a scaling factor. However, it empirically gives singular values closer to 1 than spectral normalization.

Orthogonalization techniques specifically designed for convolutional layers with circular padding have been formally derived. The most notable methods for enforcing orthogonality in circular convolutional layers include:

- Block Convolution Orthogonal Parameterization (BCOP) (Li et al., 2019) utilizes the iterative algorithm of Björck and Bowie (1971) to orthogonalize the linear transformation in convolutional layers. This transformation is applied directly to the convolutional filter K. The base implementation of BCOP is computationally expensive due to the use of block convolution.

- Skew Orthogonal Convolutions (SOC) (Singla and Feizi, 2021c) leverage the key property that the matrix exponential of a skew-symmetric matrix is always orthogonal, i.e., $\exp(\mathbf{A}) = \mathbf{W}, \quad \text{where} \quad \mathbf{A} = -\mathbf{A}^\top$. To enforce this structure in convolutional layers, SOC constructs the skew-symmetric matrix as $\mathbf{A} = \mathbf{Q}^\top - \mathbf{Q}$, where $\mathbf{Q}$ is a learnable parameter. The orthogonal transformation $\mathbf{W}$ is then obtained via a finite Taylor series approximation of the matrix exponential : it requires multiple propagations through the base convolutional layer making it computationally expensive.

- Cayley Transform (Trockman and Kolter, 2021): Orthogonalizes weight matrices via $\mathbf{W} = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{I} + \mathbf{A})$, where $\mathbf{A}$ is skew-symmetric. For convolutional layers, this transform is performed in the Fourier domain, but it requires matrix inversion, which scales with the input size.

A review of orthogonal convolutional layers is presented in (Boissin et al., 2025), highlighting both their strengths and limitations. While these approaches have demonstrated strong performance on datasets such as CIFAR-10, their scalability remains restricted to such datasets. Also, the work by Boissin et al. (2025) has significantly improved the efficiency of BCOP and SOC methods. However, enforcing equal singular values can be overly restrictive in certain cases, often requiring significantly more parameters to maintain the same level of expressivity (Béthune, 2024). Several works based on rescaling layers have been proposed in parallel and mitigate those issues.

Rescaled layers Spectral normalization (SN) (Yoshida and Miyato, 2017; Miyato et al., 2018; Farnia et al., 2019b) is a widely used technique to enforce control over the Lipschitz constant of linear layers. Unlike earlier methods, it directly leverages the Lipschitz property, offering deterministic guarantees. Specifically, weight matrices can be normalized by their largest singular values to ensure 1-Lipschitz layers (Miyato et al., 2018; Anil et al., 2019; Farnia et al., 2019a). Formally, a spectral-normalized layer is defined as:

$$\mathbf{x} \mapsto \mathbf{W}\mathbf{R}\mathbf{x},$$

where $\mathbf{R} = \frac{1}{\|\mathbf{W}\|_2}$ acts as a rescaling factor based on the spectral norm $\|\mathbf{W}\|_2$. The spectral norm is typically computed using power iteration. Throughout this work, we assume that $\|\mathbf{W}\|_2 \neq 0$ to ensure numerical stability.

During training, one iteration of Power Iteration (PI) is performed per step, with the estimated singular vector being stored in a buffer for each weight matrix. This allows for an efficient spectral norm estimation, as the weight matrices change gradually during training, making a single iteration sufficient to track the dominant singular value. This approach, introduced in Miyato et al., 2018, keeps the computational cost low while maintaining stability.

However PI can be slow to converge to the spectral norm, is non-deterministic, and does not provide a strict upper bound on the spectral norm, which can be critical in certified robustness application or normalizing flow where inversion requires a precise upper bound on the spectral norm of layers. In Chapter 3 we present a novel method for estimating spectral norms that improves upon existing techniques: faster convergence and upper bound on the spectral norm.

Another limitation of SN is stacking multiple rescaled layers with spectral normalization can lead to vanishing gradients, as all singular values except the dominant one may collapse to zero, effectively reducing the rank of the transformation and impairing gradient flow.

To address those issues, the Almost-Orthogonal Layer (AOL) (Prach and Lampert, 2022) provides a balanced approach by normalizing layers to approximate 1-Lipschitz behavior while promoting soft orthogonality. This design helps maintain stable gradient flow during training, mitigating the vanishing gradient problem observed in stacked spectral-normalized layers. The fully connected AOL layer is defined using a diagonal rescaling matrix $\mathbf{R}$, given by [1]

$$\mathbf{R} = \operatorname{diag}\left(\sum_j |\mathbf{W}^\top \mathbf{W}|_{ij}\right)^{-\frac{1}{2}}. \tag{2.5}$$

This formulation guarantees a strictly 1-Lipschitz layer, unlike spectral normalization, which only provides an approximation of the Lipschitz constant. The rescaling matrix

[1] This assumes that each column of $\mathbf{W}$ has at least one non-zero entry, ensuring that (2.5) is well-defined.

can also be computed efficiently for convolutional layers. Moreover, empirical results indicate that after training, the network's Jacobian (with respect to $\mathbf{x}$) remains nearly orthogonal, deeper architectures without excessive gradient attenuation—at least to a greater extent than spectral normalization.

Several works (Anil et al., 2019; Huang et al., 2021; Singla et al., 2022) have focused on constraining Lipschitz constants through activation functions, particularly in the setting of linear orthogonal layers. These approaches often enhance performance by maintaining a balance between orthogonality and non-linear activation behavior. More generally, designing Lipschitz layers beyond linear layers is a challenging task, as the Lipschitz constant is not directly tied to the spectral norm of the weight matrix. Wang and Manchester (2023) propose layer combinations that are inherently Lipschitz, drawing inspiration from the SDP condition of Fazlyab et al. (2019) and related works (Meunier et al., 2022; Araujo et al., 2023). They also rely on the work of Trockman and Kolter (2021) using Cayley transform to design convolutional layers. They introduce the Sandwich Layer, the transformation applied within the layer is given by

$$\mathbf{x} \mapsto \sqrt{2}\mathbf{A}^\top \mathbf{R}^{-1} \rho\left(\sqrt{2}\mathbf{R}\mathbf{B}\mathbf{x}\right),$$

where the matrices $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{B}$ are constructed as

$$\mathbf{R} = \operatorname{diag}(\exp(\mathbf{q}))^{-1}, \quad \begin{bmatrix} \mathbf{A}^\top \\ \mathbf{B}^\top \end{bmatrix} = \operatorname{Cayley}\left(\begin{bmatrix} \mathbf{W} \\ \mathbf{V} \end{bmatrix}\right),$$

with

$$\operatorname{Cayley}\left(\begin{bmatrix} \mathbf{W} \\ \mathbf{V} \end{bmatrix}\right) = \begin{bmatrix} (I + \mathbf{Z})^{-1}(I - \mathbf{Z}) \\ -2\mathbf{V}(I + \mathbf{Z})^{-1} \end{bmatrix},$$

where $\mathbf{Z} = \mathbf{W} - \mathbf{W}^\top + \mathbf{V}^\top \mathbf{V}$. Here, $\mathbf{W}$ and $\mathbf{V}$ represent the weight matrices of the layer, while $\mathbf{q}$ is a vector that parameterizes the transformation.

However, in practice, scaling very deep Lipschitz architectures remains challenging, even when stacking layers: performance is not always improved by adding more layers and the training issue can be exacerbated when the network is too deep. To mitigate this issue, residual connections can be incorporated.

Residual layers Residual connections are fundamental to modern neural network architectures, as highlighted in the seminal work of He et al. (2016). From a training dynamics perspective, residual networks are easier to optimize than plain networks, as learning residual mappings with an identity initialization is more efficient than directly learning the identity function. These connections enhance the signal-to-noise ratio and simplify the training of very deep networks by allowing information to flow directly across layers (Balduzzi et al., 2017).

The work of Meunier et al. (2022) interprets residual blocks through the lens of dynamical systems, framing residual networks as discretized continuous flows. They show that parameterizations based on the Cayley transform (Trockman and Kolter, 2021) and SOC (Singla and Feizi, 2021c) correspond to two specific discretization schemes for the skew-symmetric component of these flows. Additionally, they introduce a novel discretization approach for the symmetric component, derived from a convex potential, ensuring a 1-Lipschitz property. By discretizing this flow, they construct a new layer, the *Convex Potential Layer* (CPL), which is residual, nonlinear, and inherently 1-Lipschitz:

$$\mathbf{x} \mapsto \mathbf{x} - 2\mathbf{W}\mathbf{R}^2\rho(\mathbf{W}^\top\mathbf{x} + \mathbf{b}), \tag{2.6}$$

where $\mathbf{R} = \frac{1}{\|\mathbf{W}\|_2}$, $\|\mathbf{W}\|_2$ is the spectral norm of $\mathbf{W}$, and ρ is the ReLU activation function. This layer can also be interpreted through the lens of Maximal Monotone Operator (MMO) theory (Bauschke and Combettes, 2017), where the update corresponds to a discretized reflected resolvent step of a monotone operator.

The CPL layer is 1-Lipschitz and can be stacked to form deep networks, providing a scalable and robust alternative to standard architectures. Its bias towards the identity makes it easy to train without any additional regularization(batch norm, gradient clipping). They demonstrate successful training with up to 1000 layers on CIFAR-10 and CIFAR-100 outperforming standard architectures in terms of certified robustness. The recent survey of Prach et al. (2024) places as a good tradeoff the CPL layer in the broader context of Lipschitz networks, highlighting its potential for scalable and robust training.

Araujo et al. (2023) build upon the work of Meunier et al. (2022), their work extends this approach by developing a general framework that unifies and generalizes multiple existing Lipschitz layers, including SN, CPL, AOL, and Sandwich Layers. Specifically, they propose the SDP Lipschitz Layer (SLL), leveraging semidefinite programming (SDP) theory to ensure the 1-Lipschitz constraint while maintaining expressivity. This layer provides a structured way to derive and modify Lipschitz layers, including spectral normalization layers, AOL layers, and CPL layers. A key contribution of Araujo et al. (2023) is the introduction of a new residual layer, which retains the CPL structure but incorporates AOL-based rescaling for improved stability and flexibility. The SLL block serves as a generalized version of both CPL and AOL, where the transformation matrix $\mathbf{R}$ is adapted from AOL's construction to allow broader parameterization:

$$\mathbf{R} = \operatorname{diag}\left(\sum_{j=1}^{n} |\mathbf{W}^\top\mathbf{W}|_{ij}\frac{\mathbf{q}_j}{\mathbf{q}_i}\right)^{-\frac{1}{2}}, \tag{2.7}$$

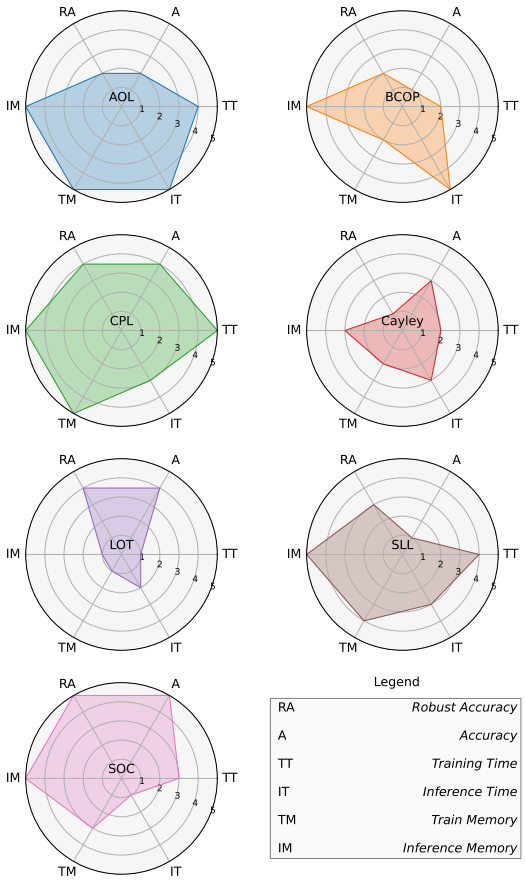


Fig. 2.2.: Figure taken from Prach et al. (2024) comparing different Lipschitz layers w.r.t to different criteria. Scores ranged from 1 (worst) to 5 (best) for every layers.

where $\mathbf{q}$ represents a vector of positive elements.

The work of Hu et al. (2023) introduced the simpler but effective *Residual Linear Block* (RLB), where the weight matrix is biased toward the identity $\mathbf{W} = \mathbf{I} + \mathbf{Q}$. They compute the spectral norm of $\mathbf{W}$ with power iteration, however, this approach does not account for nonlinearities, which are essential for capturing complex patterns in data.

All those residual architectures have converged towards a convolutional stack where the dimension of the input is preserved, and after that, a linear MLP classifier is applied. This architecture is 1-Lipschitz and has been shown to be competitive with standard architectures on various datasets (Meunier et al., 2022; Araujo et al., 2023; Hu et al., 2023; Hu et al., 2024).

Training orthogonal layers is typically computationally intensive. The Cayley method requires matrix inversion involving the image input size, which is costly, and SOC involves either singular value decomposition (SVD) or an iterative Taylor expansion. The review from Prach et al. (2024) compares the performance of different

Lipschitz layers, including BCOP, SOC, CPL, and AOL, across various criteria, such as robustness, scalability, and computational efficiency. Considering all evaluated metrics (summarized in Figure 2.2), CPL emerges as the most favorable option due to its superior performance and lower computational cost. When ample computational resources are available and strict timing constraints are not a concern during training and inference, the SOC layer may be a viable choice, given its slightly improved performance. On the other hand, applications where inference time is critical may benefit from AOL or BCOP, as they introduce no additional runtime overhead compared to standard convolution. Additionally, for higher-resolution images, CPL appears to be the most promising approach.

Limitation of Lipschitz layers Independently of the choice of layer design, the PUB is used to bound the overall Lipschitz constant of such 1-Lipschitz networks. However, the PUB is often loose because it fails to consider the interactions between layers. For deep networks, this leads to overly conservative estimates, as the product of individual Lipschitz constants inflates the bound and does not reflect the true sensitivity of the network. Additionally, decomposing neural networks into individual layers overlooks the potential smoothing effects across layers, where dependencies may naturally reduce sensitivity. Moreover, the Lipschitz property of certain non-linear layers, such as attention mechanisms, can be challenging to determine Kim et al., 2021. To address those limitations we are going to present the Weierstrass transform in the next section, which is a smoothing technique that can be used to regularize neural networks and improve their Lipschitz properties.

2.1.4 Weierstrass transform

To address limitations of PUB, others techniques exist to improve stability and robustness. One such method is the *Weierstrass transform* (Zayed, 1996), a classical smoothing technique from functional analysis. Also known as Gaussian convolution, it consists in convolving a function with a Gaussian kernel, the Weierstrass transform enhances regularity, making the function continuous and differentiable. This smoothing effect not only mitigates sensitivity to perturbations but also induces global regularity by smoothing decision boundaries. As stated in Salman et al. (2019), *randomized smoothing* (RS) (Lecuyer et al., 2018; Cohen et al., 2019) is an application of the Weierstrass transform in the context of neural networks leveraging its probabilistic interpretation. Originating from differential privacy, RS derives its certified robustness guarantees from a probabilistic perspective, as introduced by Lecuyer et al. (2018). It applies the principle of convolution with a predefined probability distribution to the input, commonly relying on Gaussian perturbations, enabling scalability to large datasets like ImageNet. In practice, it relies commonly

on Gaussian perturbations on the input rather than transformations of the entire network.

Definition 2.1.3 (Weierstrass transform (Zayed, 1996))**.**
Let $f : \mathbb{R}^d \to \mathbb{R}$ be a measurable function integrable with respect to the Gaussian kernel. The Weierstrass transform of f is defined as:

$$\tilde{f}(\mathbf{x}) = (f * \phi_\sigma)(\mathbf{x}) = \int_{\mathbb{R}^d} f(\mathbf{x} - \delta)\, \phi_\sigma(\delta)\, d\delta,$$

where $\phi_\sigma(\delta) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\delta\|^2}{2\sigma^2}\right)$ is the isotropic Gaussian kernel. Since ϕ_σ is symmetric, this expression is equivalent to

$$\tilde{f}(\mathbf{x}) = \int_{\mathbb{R}^d} f(\mathbf{x} + \delta)\, \phi_\sigma(\delta)\, d\delta,$$

which corresponds to the probabilistic interpretation used in randomized smoothing.

In the following, we always assume that f is measurable and that the integral defining $\tilde{f}$ is well-defined.

Alternatively, the Weierstrass transform can be interpreted probabilistically. Let $\delta \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ be a random vector following a multivariate Gaussian distribution with mean zero and covariance matrix $\sigma^2\mathbf{I}$. Then, the Weierstrass transform is the expectation of f over this Gaussian distribution:

$$\tilde{f}(\mathbf{x}) = \mathbb{E}_{\delta \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})} \left[f(\mathbf{x} + \delta) \right].$$

This probabilistic interpretation highlights that the Weierstrass transform smooths f by averaging its values over Gaussian perturbations of the input $\mathbf{x}$. When the integral is not explicitly computable, this allows its estimation through Monte Carlo integration by sampling from the Gaussian distribution as done in RS. This transform smooths f by averaging its values over neighborhoods determined by the parameter σ, enhancing the regularity of f and making $\tilde{f}$ infinitely differentiable (C^∞) because the Gaussian kernel allows differentiation under the integral sign (Stein, 1970).

Lemma 2.1.4 (Stein's Lemma (Stein, 1970))**.**
Let $\sigma > 0$, and let $f : \mathbb{R}^d \to \mathbb{R}$ be a function. Then $\tilde{f}$ is differentiable, and its gradient is given by:

$$\nabla \tilde{f}(\mathbf{x}) = \frac{1}{\sigma^2} \mathbb{E}_{\delta \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})} \left[\delta f(\mathbf{x} + \delta) \right].$$

Using Stein's Lemma, we can further characterize the regularity properties of $\tilde{f}$ under an additional boundedness assumption.

Lemma 2.1.5 (Lipschitz continuity of the Weierstrass transform for bounded functions (Salman et al., 2019))**.**
Let $\sigma > 0$, and let $f : \mathbb{R}^d \to \mathbb{R}$ be a function satisfying $|f(\mathbf{x})| \leq 1$ for all $\mathbf{x} \in \mathbb{R}^d$. Then, $\tilde{f}$ is Lipschitz continuous with respect to the ℓ_2-norm, and :

$$\mathrm{L}\left(\tilde{f}\right) \leq \sqrt{\frac{2}{\pi\sigma^2}}.$$

Note that when the function f is Lipschitz continuous, the Weierstrass transform $\tilde{f}$ inherits this property, as shown in the following lemma:

Lemma 2.1.6 (Lipschitz continuity of the Weierstrass transform for Lipschitz functions (Nesterov and Spokoiny, 2017))**.**
Let $f : \mathbb{R}^d \to \mathbb{R}$ be a Lipschitz continuous function with Lipschitz constant $\mathrm{L}(f)$. Then $\widetilde{f}$ is Lipschitz continuous with respect to the ℓ_2-norm and its Lipschitz constant is given by: $\mathrm{L}\left(\widetilde{f}\right) \leq \mathrm{L}(f)$.

Coming from randomized smoothing Salman et al. (2019), a stronger Lipschitz bound can be derived by considering the composition of quantile function with a smoothed classifier. This involves considering the Lipschitz constant of the following function:

$$(\Phi^{-1} \circ \tilde{f})(\mathbf{x}) = \Phi^{-1}\left(\mathbb{E}_{\delta \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})}\left[f(\mathbf{x} + \delta)\right]\right),$$

where Φ^{-1} is the Gaussian quantile, defined as the inverse of the Gaussian cumulative distribution function. The function $\Phi^{-1} \circ \tilde{f}$ is Lipschitz continuous with respect to the ℓ_2-norm, with its Lipschitz constant given by:

Lemma 2.1.7 (Lipschitz bound for quantile-composed smoothed classifier (Cohen et al., 2019; Salman et al., 2019))**.**
$\Phi^{-1} \circ \tilde{f}$ is Lipschitz continuous with respect to the ℓ_2-norm, and its Lipschitz constant is given by:

$$\mathrm{L}\left(\Phi^{-1} \circ \tilde{f}\right) = \frac{1}{\sigma}.$$

If we suppose that the function f is Lipschitz continuous it is not straightforward to derive a bound on the Lipschitz constant of the quantile-composed smoothed classifier which intervenes $\mathrm{L}(f)$. Indeed, the quantile function is not Lipschitz

continuous, and the composition of a Lipschitz function with a non-Lipschitz function does not guarantee a Lipschitz function.

In the next section, we are going to see how Lipschitz networks can be used to provide certification in the context of adversarial robustness.

2.2 Certified robustness

Consider a classification problem where the input space is $\mathcal{X} \subset \mathbb{R}^d$, and the label space is $\mathcal{Y} = \{1, \dots, c\}$, representing c distinct classes. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_{\text{train}}}$ denote a dataset of n_{train} labeled examples, where each $\mathbf{x}_i \in \mathcal{X}$ is an input, and $y_i \in \mathcal{Y}$ is its corresponding label. A function $f : \mathcal{X} \to \mathbb{R}^c$ maps each input $\mathbf{x} \in \mathcal{X}$ to a vector of scores $f(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_c(\mathbf{x})]^\top$, where $f_k(\mathbf{x})$ represents the predicted confidence for class k. The final classifier is defined as:

$$\hat{y} = \arg\max_k f_k(\mathbf{x}),$$

where $\hat{y}$ is the predicted label. $f_y(\mathbf{x})$ is the score for the true label y of the input $\mathbf{x}$.

In a deep learning context, f is parameterized as a neural network, trained on a labeled dataset to minimize a classification loss. When trained effectively, f achieves high accuracy on test data, successfully predicting the correct label y for most inputs $\mathbf{x}$ (Krizhevsky et al., 2012). Despite this success, deep neural networks are highly vulnerable to adversarial attacks (Szegedy et al., 2013; Goodfellow et al., 2015), where small, carefully designed perturbations to the input can cause incorrect predictions. This instability raises concerns in critical applications such as autonomous driving and healthcare. In Figure 2.3, we illustrate an adversarial attack on an image classifier, where an imperceptible perturbation causes the classifier to misclassify an image of a pig as an airliner.

Definition 2.2.1 (Adversarial attacks (Szegedy et al., 2013))**.**
Let $\mathbf{x} \in \mathcal{X}$ be an input, $y \in \mathcal{Y}$ its true label, and f a classifier. An adversarial perturbation at level ε is a vector η such that $\|\eta\| \leq \varepsilon$ and:

$$\arg\max_k f_k(\mathbf{x} + \eta) \neq y.$$

The level ϵ is also called adversarial budget, and it quantifies the maximum perturbation allowed to the input. In this work and commonly in the literature we evaluate adversarial (Carlini and Wagner, 2017; Madry et al., 2018) robustness

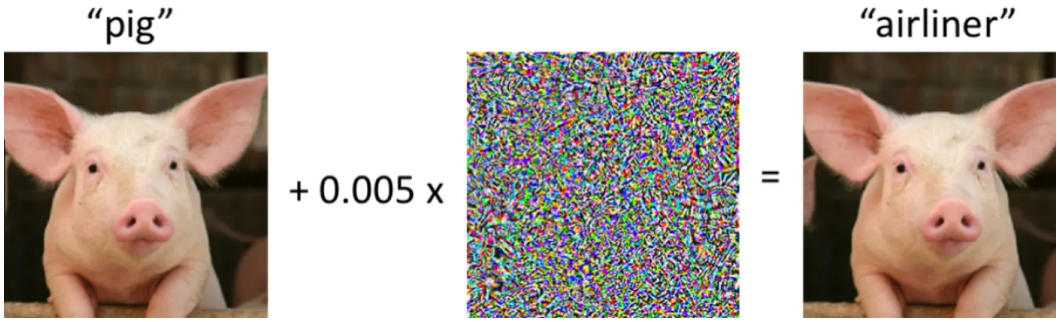


Fig. 2.3.: Example of an adversarial attack on an image classifier. The original image (left) is correctly classified as a pig, but the perturbed image (right) is misclassified as an airliner. The perturbation is imperceptible to the human eye but causes the classifier to make an incorrect prediction. The added noise is scaled by a factor of 0.005 for visualization purposes. Example was taken from the blogpost of Haldar (2020).

under ℓ_2-bounded perturbations η such that $\|\eta\|_2 \leq \varepsilon$, where ε is expressed in pixel scale units, i.e., relative to the input domain $[0, 255]$ (image range pixel intensity). In normalized input space $[0, 1]$, this corresponds to $\varepsilon_{\text{normalized}} = \frac{\varepsilon_{\text{raw}}}{255}$.

To construct and simple and effective adversarial attack one can use the Projected Gradient Descent (PGD) attack (Madry et al., 2018). The PGD attack iteratively updates an adversarial perturbation η to maximize the classifier's loss while ensuring the perturbation remains within the allowed ℓ_2-norm constraint, see Algorithm 1. Several methods have been proposed to enhance the robustness of classifiers. Adver-

Algorithm 1 PGD Attack under ℓ_2-norm Constraint

1: **Input**: Classifier f, input $\mathbf{x}$, true label y, step size α, perturbation budget ϵ, number of iterations T.
2: **Initialize**: $\eta_0 \leftarrow$ random unit vector $\cdot\, \epsilon$
3: **for** $t = 0$ to $T - 1$
4: Compute gradient: $g \leftarrow \nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x} + \eta_t), y)$
5: Normalize gradient: $g \leftarrow g / \|g\|_2$
6: Update perturbation: $\eta_{t+1} \leftarrow \eta_t + \alpha g$
7: Project onto ℓ_2-ball: $\eta_{t+1} \leftarrow \epsilon \cdot \frac{\eta_{t+1}}{\max(\epsilon, \|\eta_{t+1}\|_2)}$
8: **Return** $\mathbf{x}_{\text{adv}} = \mathbf{x} + \eta_T$

sarial training (Madry et al., 2018) strengthens models by incorporating adversarial examples into the training process. However, the ongoing interplay between attacks and defenses has been likened to a *cat-and-mouse game*, where new attacks continually challenge existing defenses (Athalye et al., 2018). This dynamic nature highlights the need for defenses with formal, certified guarantees of robustness. Certified defenses, such as Lipschitz continuity (Cisse et al., 2017; Tsuzuku et al., 2018) and *randomized smoothing* (RS) (Lecuyer et al., 2018; Li et al., 2018a; Cohen et al., 2019), have emerged to address the challenges posed by adversarial attacks. A key metric for evaluating these defenses is the *certified robust radius*, which quanti-

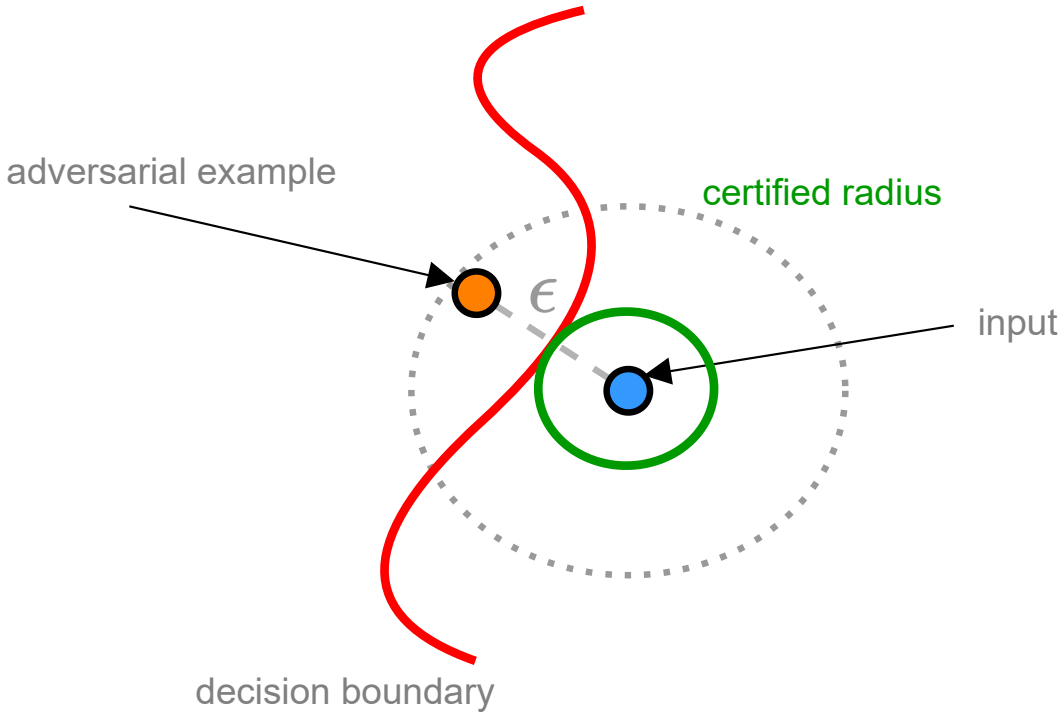


Fig. 2.4.: Illustration of an adversarial attack $\mathbf{x}_{\text{adv}}$ of budget ϵ on an input $\mathbf{x}$ with classifier boundaries and the certified radius in green. No adversarial perturbation within the certified radius can cause the classifier to change its decision.

fies the maximum allowable perturbation to an input $\mathbf{x}$ that ensures the classifier's decision remains stable. Larger certified radii indicate greater robustness.

Definition 2.2.2 (Certified radius (Tsuzuku et al., 2018))**.**
For a classifier $f : \mathcal{X} \to \mathbb{R}^c$, an input $\mathbf{x} \in \mathcal{X}$, and its label y, the certified radius $R(f, \mathbf{x}, y)$ is defined as:

$$\mathrm{R}(f, \mathbf{x}, y) := \inf \left\{ \epsilon \mid \epsilon > 0,\ \exists \eta \in B_2(\mathbf{0}, \epsilon),\ \arg\max_k f_k(\mathbf{x} + \eta) \neq \arg\max_k f_k(\mathbf{x}) \right\}.$$

where $B_2(\mathbf{0}, \epsilon) = \{\eta \in \mathbb{R}^d \mid \|\eta\|_2 \leq \epsilon\}$.

Figure 2.4 illustrates the concept of certified robustness, where the certified radius quantifies the maximum allowable perturbation ϵ to an input $\mathbf{x}$ that ensures the classifier's decision remains stable. To measure the robustness of a classifier, we define the *certified accuracy* as the proportion of inputs for which the classifier is guaranteed to be robust against any perturbation within a given radius ϵ. The certified accuracy provides a quantitative measure of the classifier's robustness, reflecting the proportion of inputs for which the classifier's decision remains stable under perturbations.

Definition 2.2.3 (Certified accuracy for ℓ_2-norm perturbations (Tsuzuku et al., 2018))**.**
The certified accuracy *of a classifier at a perturbation radius ϵ is the proportion of inputs for which the classifier is guaranteed to be robust against any perturbation within the ℓ_2-norm ball of radius ϵ. The certified accuracy at radius ϵ is defined as:*

$$\frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{1}_{(\arg\max_k f_k(\mathbf{x}+\eta)=y,\ \forall\ \|\eta\|_2\leq\epsilon)} \cdot$$

2.2.1 Lipschitz bounded networks

The concept of Lipschitz continuity complements the certified robust radius by introducing the Lipschitz constant, which quantifies the sensitivity of the network f to input perturbations η. Specifically, a function f is Lipschitz continuous if there exists a constant $\mathrm{L}(f) > 0$ such that:

$$\|f(\mathbf{x}+\eta) - f(\mathbf{x})\| \leq \mathrm{L}(f)\|\eta\|.$$

A smaller Lipschitz constant implies that the network f exhibits slower variations in its output with respect to changes in its input, thus reducing the risk of instability under perturbations. In parallel, the prediction margin,

$$\mathrm{M}(f(\mathbf{x}), y) = \max(0, f_y(\mathbf{x}) - \max_{k\neq y} f_k(\mathbf{x})),$$

quantifies the confidence of the classifier f in assigning the label y to the input $\mathbf{x}$. A larger margin indicates greater confidence and resilience to perturbations. Together, the Lipschitz constant and the margin are interconnected quantities that directly influence the certified robust radius $\mathrm{R}(f, \mathbf{x}, y)$.

A simple lower bound on the certified radius can be derived from the Lipschitz constant of the network. This bound is expressed as:

$$\mathrm{R}(f, \mathbf{x}, y) \geq \min_{y'\neq y} \frac{\max(f_y(\mathbf{x}) - f_{y'}(\mathbf{x}), 0)}{\mathrm{L}(f_y - f_{y'})}. \tag{2.8}$$

The Lipschitz constant $\mathrm{L}(f_y - f_{y'})$ can be bounded by $\mathrm{L}(f_y) + \mathrm{L}(f_{y'})$, giving the expression:

$$\min_{y'\neq y} \frac{\max(f_y(\mathbf{x}) - f_{y'}(\mathbf{x}), 0)}{\mathrm{L}(f_y) + \mathrm{L}(f_{y'})}.$$

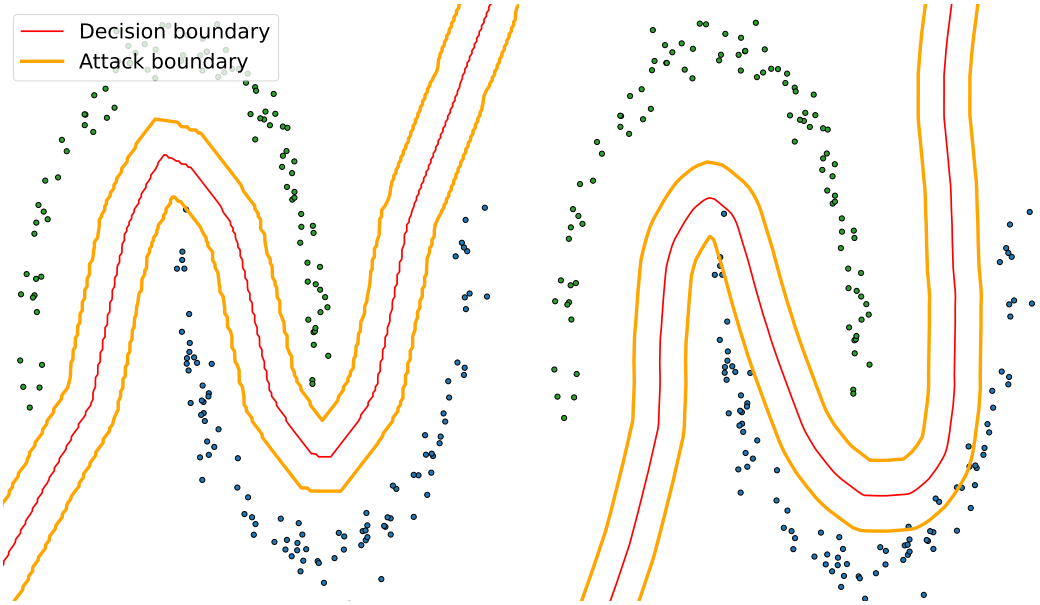


Fig. 2.5.: Illustration of the attack perimeter (orange) for a given attack budget for a MLP classifier with two classes (green and blue dots). The margin is the distance between the decision boundary (red) and the closest point to the decision boundary. The left figure shows the decision boundary with larger margins than the right figure ensuring robustness to the attack perimeter in orange.

This bound reflects the sensitivity of individual scores to input perturbations. When all logits share the same Lipschitz constant, robustness depends on the logit margin scaled by this sensitivity:

$$\mathrm{R}_{\mathrm{coord}}(f, \mathbf{x}, y) = \frac{\mathrm{M}(f(\mathbf{x}), y)}{2\mathrm{L}(f_y)}. \tag{2.9}$$

Since robustness is determined by the smoothness of each logit rather than the entire classifier, this justifies the term coordinate-wise bound and derived radius $\mathrm{R}_{\mathrm{coord}}$.

This expression shows that the certified radius is directly proportional to the margin and inversely proportional to the Lipschitz constant. Larger margins and smaller Lipschitz constants lead to greater robustness. In Figure 2.5, we illustrate the attack perimeter for a given budget and the same Lipschitz constant for both classifiers, illustrating that larger margins increase robustness. Note that both classifiers perform with perfect accuracy on the test set.

For the ℓ_2-norm, Tsuzuku et al., 2018 derived a tighter bound for $\mathrm{L}(f_y - f_{y'})$.

Proposition 2.2.4 (Lipschitz-margin lower bound on the certified radius (Tsuzuku et al., 2018))**.**

Given a Lipschitz continuous network f under the ℓ_2-norm, and given a perturbation level $\varepsilon > 0$, $\mathbf{x} \in \mathcal{X}$, and $y \in \mathcal{Y}$ as the label of $\mathbf{x}$, if the margin satisfies:

$$\mathrm{M}(f(\mathbf{x}), y) > \sqrt{2}\mathrm{L}(f)\varepsilon,$$

then for every η such that $\|\eta\|_2 \leq \varepsilon$, we have $\arg\max_k f_k(\mathbf{x} + \eta) = y$.

Reworking this proposition, a lower bound for the certified radius is given by

$$\mathrm{R}_{\mathrm{global}}(f, \mathbf{x}, y) = \frac{\mathrm{M}(f(\mathbf{x}), y)}{\sqrt{2}\mathrm{L}(f)}. \tag{2.10}$$

Since it relies on the global Lipschitz property, it is referred to as the $\mathrm{R}_{\mathrm{global}}$. Indeed, for Lipschitz-by-design networks, $\mathrm{L}(f)$ can be bounded explicitly, making it straightforward to estimate the certified radius by evaluating the margin at the input $\mathbf{x}$. These results highlight the critical role of Lipschitz continuity in achieving certified robustness, demonstrating that controlling $\mathrm{L}(f)$ and maximizing the margin $\mathrm{M}(f(\mathbf{x}), y)$ are key strategies for designing robust models.

2.2.2 Randomized smoothing

Consider the set $\Delta^{c-1} = \{\mathbf{p} \in \mathbb{R}^c_+ \mid \mathbf{1}^\top \mathbf{p} = 1\}$ defines the $(c-1)$-dimensional probability simplex. Let $\tau : \mathbb{R}^c \mapsto \Delta^{c-1}$ represent a mapping onto this simplex, typically corresponding to functions like $\mathrm{softmax}$ or $\mathrm{hardmax}$. For a logit vector $\mathbf{z} \in \mathbb{R}^c$, the mapping onto Δ^{c-1} is denoted by $\tau(\mathbf{z})$. A specific case of this mapping is the $\mathrm{hardmax}$, where for each component k, we have $\tau_k(\mathbf{z}) = \mathbb{1}_{\arg\max_i \mathbf{z}_i = k}$.

We note $f : \mathcal{X} \mapsto \mathbb{R}^c$ as the network, which produces the logits before applying τ. $\mathrm{hardmax}$ is a common choice for τ in the context of randomized smoothing, as it assigns all the probability mass to the highest logit, softmax is also used in the work of Levine et al. (2019). The soft classifier $F : \mathbb{R}^d \to \Delta^{c-1}$ is defined as:

$$F(\mathbf{x}) = \tau(f(\mathbf{x})) = [F_1(\mathbf{x}), \ldots, F_c(\mathbf{x})]^\top, \tag{2.11}$$

which outputs a probability distribution over the c classes. The final decision is given by the hard classifier $F^{\mathrm{H}} : \mathbb{R}^d \to \mathcal{Y}$, defined as:

$$F^{\mathrm{H}}(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x}),$$

which returns the predicted label $\hat{y} = F^{\mathrm{H}}(\mathbf{x})$.

Randomized smoothing (RS) was initially proposed by Li et al., 2018b; Lecuyer et al., 2019 and further developed in key works (Cohen et al., 2019; Salman et al., 2019). The core idea behind RS is to improve the robustness of a classifier by averaging its predictions over Gaussian perturbations of the input. This gives a hard smoothed classifier $\tilde{F}^H$, which is more resistant to adversarial attacks than the classifier F^H. The soft smoothed classifier $\tilde{F}$ is defined as the Weierstrass transform of F:

$$\tilde{F}(\mathbf{x}) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2\mathbf{I})}\left[F(\mathbf{x}+\delta)\right] \ .$$

This means that $\widetilde{F}^{\mathrm{H}} = \arg\max_k \tilde{F}_k(\mathbf{x})$ returns the class k that has the highest probability when Gaussian noise δ is added to the input $\mathbf{x}$.

Note that if $\tau = \mathrm{hardmax}$, then for all $k \in \mathcal{Y}$, we have:

$$\tilde{F}_k(\mathbf{x}) \ = \ \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I)}\Big[\arg\max_i \ f_i(\mathbf{x}+\delta) \ = \ k\Big],$$

which is the usual formulation in the literature (Cohen et al., 2019) based on the probabilistic interpretation of the smoothed classifier. This quantity corresponds to the probability that class k wins the majority vote under Gaussian perturbations. However, we retain the more general notation to accommodate other simplex mappings τ besides $\mathrm{hardmax}$ in the remainder of this thesis. For ease of notation, let's denote $\mathbf{p} = \widetilde{F}(\mathbf{x})$, the vector of output probabilities given by the smoothed classifier, with $\mathbf{p}_k$ its k–th component. A key contribution of RS is that it allows us to compute a lower bound on the certified radius. Its estimation is based on the top two class probabilities, denoted as $\mathbf{p}_{i_1}$ and $\mathbf{p}_{i_2}$ where $i_1 = \arg\max_k \mathbf{p}_k(\mathbf{x})$ and $i_2 = \arg\max_{k \neq i_1} \mathbf{p}_k(x)$. In practice the lower bound on the radius is obtained using upper bound $\overline{\mathbf{p}}_{i_2}$ and respectively lower bound ($\underline{\mathbf{p}}_{i_1}$) on the probabilities $\mathbf{p}_{i_2}$ and respectively $\mathbf{p}_{i_1}$. The Figure 2.6 illustrates the decision regions of the base classifier F and the probability distribution of $F(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}))$ for a given input $\mathbf{x}$. Two main approaches have been proposed to derive the certified radius. The first relies on a probabilistic proof using the Neyman-Pearson lemma (Cohen et al., 2019). The second employs a deterministic proof based on the Weierstrass transform regularization to enforce Lipschitz continuity, which is then used with the coordinate-wise certified radius $\mathrm{R}_{\mathrm{coord}}$ (Salman et al., 2019). The bound on certified radius is given in the following Theorem 2.2.5:

Theorem 2.2.5 (Certified radius for randomized smoothing (Cohen et al., 2019)). *Let $f : \mathbb{R}^d \to \mathbb{R}^c$ be any deterministic or random function, its associated soft classifier $F = \mathrm{hardmax} \circ f$, and let $\delta \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Define the smoothed classifier $\tilde{F}$ as:*

$$\tilde{F}(\mathbf{x}) = \mathbb{E}(F(\mathbf{x}+\delta)).$$

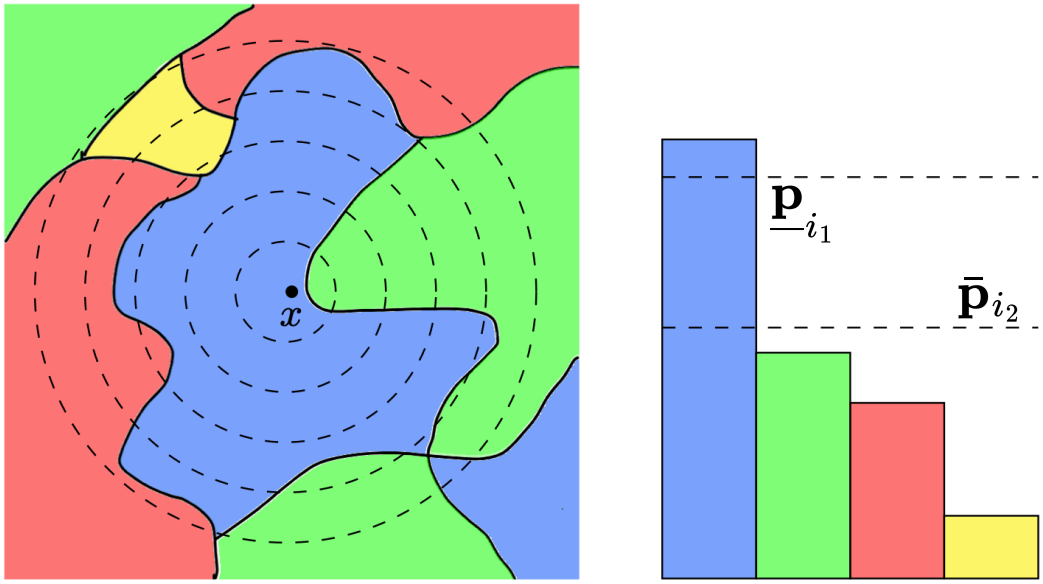


Fig. 2.6.: Evaluating the smoothed classifier at a given input $\mathbf{x}$. Left: the decision regions of the base classifier F are shown in different colors. The dotted lines represent the level sets of the Gaussian distribution $\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$. Right: the probability distribution of $F(\mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}))$. Here, $\underline{\mathbf{p}}_{i_1}$ is a lower bound on the probability of the predicted (top) class, and $\overline{\mathbf{p}}_{i_2}$ is an upper bound on the probability of each other class. In this example, the smoothed classifier assigns the input to the "blue" region. Figure taken from Cohen et al., 2019

.

We note $\mathbf{p} = \tilde{F}(\mathbf{x})$*, suppose* $i_1 \in \mathcal{Y}$ *and* $\underline{\mathbf{p}}_{i_1}, \overline{\mathbf{p}}_{i_2} \in [0, 1]$ *satisfy:*

$$\mathbf{p}_{i_1} \geq \underline{\mathbf{p}}_{i_1} \geq \overline{\mathbf{p}}_{i_2} \geq \mathbf{p}_{i_2}.$$

Then, $\tilde{F}^{\mathrm{H}}(\mathbf{x} + \delta) = i_1$ *for all* $\|\eta\|_2 < \mathrm{R}$*, where the certified radius* R *is given by:*

$$\mathrm{R} = \frac{\sigma}{2}\left(\Phi^{-1}(\underline{\mathbf{p}}_{i_1}) - \Phi^{-1}(\overline{\mathbf{p}}_{i_2})\right), \tag{2.12}$$

and Φ^{-1} *denotes the inverse cumulative distribution function of the standard Gaussian distribution.*

In practice most works (Cohen et al., 2019; Salman et al., 2019; Yang et al., 2020a) used $\underline{\mathbf{p}}_{i_1} = \mathbf{p}_{i_1}$ and $\overline{\mathbf{p}}_{i_2} = 1 - \mathbf{p}_{i_1}$ to simplify the computation of the certified radius. Resulting in the following mono-class certified radius, smaller than Equation 2.12:

$$\mathrm{R}_{\mathrm{mono}}(\mathbf{p}) = \sigma\Phi^{-1}(\mathbf{p}_{i_1}) \leq \mathrm{R}. \tag{2.13}$$

This mono-class certified radius $\mathrm{R}_{\mathrm{mono}}$ is non trivial only if the winning class probability $\mathbf{p}_{i_1}$ is greater than $\frac{1}{2}$, otherwise it is equal to zero.

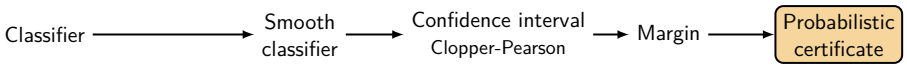


Fig. 2.7.: Cohen et al., 2019 smooths soft classifier F to create the soft smoothed classifier $\tilde{F}$. The risk factor α is then estimated using the Clopper-Pearson interval to provide a probabilistic certificate using the Neyman-Pearson lemma.

This radius provides a quantifiable measure of the classifier's robustness. The probabilities $\mathbf{p}$ are estimated using a Monte Carlo (MC) approach. Given n Gaussian samples $\delta_i \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, the empirical estimate of $\mathbf{p}$ is:

$$\hat{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} F(\mathbf{x} + \delta_i) \ .$$

Since the certified radius R is derived from the probabilities $\mathbf{p}_{i_1}$ (for the winning class) and $\mathbf{p}_{i_2}$ (for any other class), we estimate these values using Monte Carlo sampling. This estimation introduces uncertainty due to the limited number of samples. To control this uncertainty and to compute R at an exact risk level α, confidence interval methods are employed. In particular, lower and upper bounds, $\underline{\mathbf{p}}_{i_1}$ and $\overline{\mathbf{p}}_{i_2}$, are obtained via concentration inequalities (see, e.g., Lecuyer et al., 2019; Levine et al., 2019) or the Clopper-Pearson method (as in Cohen et al., 2019). These bounds provide a conservative margin that is then converted into the certified radius R. The overall randomized smoothing process is summarized in Figure 2.7.

Randomized smoothing relies on injecting noise at inference time to improve the robustness of deep neural networks. Notably, randomized smoothing can be interpreted as a form of regularization, as it aims to control the model's sensitivity to input perturbations. In the next section, we explore regularization techniques that explicitly enforce smoothness for improved generalization.

2.3 Regularization for generalization

Regularization techniques are essential for enhancing the generalization capability of neural networks by controlling model complexity, reducing overfitting, and stabilizing training. Generalization is the ability of a model to perform well on unseen data. It is often associated with the Lipschitz constant of the network (Bartlett et al., 2017) and the flatness of the minima found during training (Hochreiter and Schmidhuber, 1997a) as highlighted by empirical studies (Keskar et al., 2017; Chaudhari et al., 2019), where sharp minima are linked to poorer generalization.

For a given label $\mathbf{y}$, the loss is denoted by $\mathcal{L}(\mathbf{y}, \boldsymbol{\theta})$ in short, where $\boldsymbol{\theta} = \mathrm{vec}\left(\{\mathbf{W}^{(l)}\}_{l=1,\dots,L}\right)$. For brevity, we write the loss function as $\mathcal{L}(\theta)$, and assuming it is differentiable, we denote respectively $\nabla_\theta \mathcal{L}$ and $\nabla^2_\theta \mathcal{L}$ as its gradient and Hessian with respect to the parameters θ.

2.3.1 Lipschitz constant

Lipschitz constant regularization has emerged as a significant technique for improving the generalization capabilities of deep neural networks. By constraining the Lipschitz constant, these methods aim to control the sensitivity of the model to input perturbations, leading to smoother decision boundaries and enhanced robustness. Finlay et al. (2018) demonstrated that input gradient regularization, which implicitly enforces a Lipschitz constraint, results in improved generalization and adversarial robustness. The work of Yoshida and Miyato (2017) introduces a regularization scheme that penalizes the sum of the spectral norms of weight matrices by adding a corresponding term to the loss function.

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) + \lambda \sum_{l=1}^{L} \|\mathbf{W}_l\|_2,$$

where λ is a hyperparameter controlling the strength of the regularization term. In the same flavor Gouk et al. (2021) introduced a projection method to enforce Lipschitz continuity in neural networks for different norms. Instead of relying on regularization, they enforce a hard constraint. Furthermore, Virmaux and Scaman (2018) provided an analysis of Lipschitz continuity in deep neural networks and proposed an efficient estimation technique, contributing to the understanding of how Lipschitz constraints can be applied to enhance generalization.

2.3.2 Flat minima

The relationship between loss landscapes, generalization, and stochastic gradient descent (SGD) has been a central topic in machine learning research for years (Hochreiter and Schmidhuber, 1997a). For instance, it has been shown that in overparameterized models, local minima of the loss function are often close to the global minima (Choromanska et al., 2015). Additionally, Xing et al. (2018) demonstrates that SGD exhibits an implicit bias, favoring regions of the loss landscape resembling a valley. The sharpness is captured by the magnitude of the singular values of the Hessian of the loss w.r.t parameters, $\|\nabla^2 \mathcal{L}(\boldsymbol{\theta})\|_2$ (worst sharpness) or $\mathrm{Tr}(\nabla^2 \mathcal{L}(\boldsymbol{\theta}))$ (average sharpness) and reflects the curvature of the loss landscape. A pivotal study by Keskar et al. (2017) shows that large-batch training tends to converge to sharp minima, which correlates with worse generalization compared to the flatter minima

achieved with small-batch training, as Hessian measure are hard to compute in practice, other metrics have been proposed to assess sharpness, such as

$$\frac{\max_{\boldsymbol{\Delta}\epsilon\mathcal{B}} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{\Delta}) - \mathcal{L}(\boldsymbol{\theta})}{1+\mathcal{L}(\boldsymbol{\theta})} , \tag{2.14}$$

where $\mathcal{B}$ is a ball around the current parameters $\boldsymbol{\theta}$. However, Dinh et al. (2017) point out that common sharpness metrics, such as the spectral norm of the Hessian of $\mathcal{L}(\boldsymbol{\theta})$ and proxies, are sensitive to re-scaling. It is important to note that flatter minima do not always guarantee better generalization. Counterexamples exist where sharp minima generalize well and flat minima perform poorly (Zhang et al., 2017; Andriushchenko and Flammarion, 2022). Nevertheless, Jiang* et al. (2020) show that sharpness-based metrics often outperform other complexity metrics for evaluating generalization, such as the metric from Equation 2.14. Sharpness-Aware Minimization (SAM), introduced by Foret et al. (2021), is a widely studied loss smoothing technique that explicitly promotes flat minima by penalizing sharp regions of the loss landscape. SAM minimizes a robust smoothed objective:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\Delta}\epsilon\mathcal{B}(0,\rho)} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{\Delta}) , \tag{2.15}$$

where $\mathcal{B}(0,\rho)$ represents a ball of radius ρ around zero, and $\boldsymbol{\Delta}$ is the perturbation applied to the parameters. By introducing an adversarial component into the optimization process, SAM effectively reduces sharpness in the loss landscape, which improves generalization across many computer vision tasks. Flat minima, known to enhance generalization, can be achieved through various regularization strategies, including noise injection.

2.3.3 Loss smoothing and noise injection

Noise injection methods add stochasticity during training, either to inputs, activations, or weights. It improves robustness and prevents overfitting. The most widely known is probably the dropout (Srivastava et al., 2014), which randomly drops units during training, reducing co-adaptation of neurons and improving generalization, it is applied through noise injection on activations. However, its stochastic nature introduces additional variance in performance, requiring careful tuning to prevent underfitting, particularly when applied to deep models (Liu et al., 2023). The increased training variability may also result in unstable training dynamics under certain conditions. Perturbed Gradient Descent (PGD) (Jin et al., 2017) introduces noise into the weight updates to help models escape saddle points and better explore the optimization landscape. Building on PGD, (Orvieto et al., 2022) modifies the noise structure and proves it promotes convergence to flatter minima by controlling

the curvature of the loss landscape. The resulting optimization program can be summarized with the smoothed loss:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\Delta} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})} \left[\mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\Delta}) \right].$$

The resulting optimization landscape becomes smoother, reducing the impact of sharp regions. However, tuning the noise parameter σ is critical, as improper choices can lead to instability with exploding variance or insufficient regularization.

While noise-based regularization techniques have been widely adopted, they often introduce undesirable variance into the training process, which can hinder performance. The work of Bishop (1995) interprets Gaussian noise on inputs as Tikhonov regularization on parameters also known as weight decay, establishing a connection between noise injection and deterministic loss smoothing. This encourages smaller weights and promotes simpler solutions that generalize better (Krogh and Hertz, 1992). Such regularization flattens the loss landscape, leading to convergence toward wider, flatter minima that correlate with improved generalization (Hochreiter and Schmidhuber, 1997a). The regularized loss function can be expressed as:

$$\min_{\theta} \mathcal{L}(\boldsymbol{\theta}) + \frac{\sigma^2}{2} \|\boldsymbol{\theta}\|_2^2.$$

Notable other works proposed layer-wise regularization such as margin constraint (Elsayed et al., 2018) or activation regularization (Yashwanth et al., 2024).

2.4 Conclusion

In this chapter, we provided an overview of key concepts in neural network robustness and generalization, focusing on Lipschitz continuity, certified robustness, and regularization techniques. First, we introduced Lipschitz continuity as a fundamental property of neural networks, impacting stability, adversarial robustness, and generalization. We examined methods for estimating the Lipschitz constant, including product upper bounds and local Lipschitz estimation techniques. Additionally, we discussed architectural constraints, such as orthogonal, rescaled, and residual layers, designed to enforce Lipschitz constraints explicitly. We then explored certified robustness, highlighting how Lipschitz-bounded networks provide deterministic certification guarantees, while randomized smoothing offers a probabilistic alternative. We established the connection between the Weierstrass transform and randomized smoothing, showing how both techniques introduce controlled smoothing to certify robustness. Finally, we examined regularization techniques for improving generalization, including Lipschitz-based constraints, flat minima optimization, and

noise-based methods such as dropout and loss smoothing. These approaches aim to control model complexity, reduce overfitting, and improve stability during training.

Overall, this chapter established the theoretical foundations necessary for understanding stability in deep learning, with a focus on Lipschitz continuity. In the following chapters, we delve deeper into practical implementations of Lipschitz networks, loss smoothing techniques, and their applications to certified robustness and generalization. A key challenge in Lipschitz constant estimation, Lipschitz regularization, and Lipschitz layer design is the computation of the spectral norm for linear layers, which is crucial for the implementation of the methods presented in the subsequent chapters. To address this, we introduce in the next chapter Gram iteration, a novel approach for spectral norm estimation that is faster, deterministic, and provides strict upper bound guarantees for both convolutional and dense layers.

3 Spectral norm computation

Contents

The spectral norm of a matrix, defined as its largest singular value, is a fundamental quantity in numerical linear algebra, optimization, and control theory (Golub and Loan, 2012). Accurate and efficient computation of the spectral norm is essential for tasks ranging from stability analysis and system design to matrix approximations and data regularization (Lanczos, 1950; Arnoldi, 1951; Kublanovskaya, 1962; Lehoucq et al., 1996). In deep learning, it plays a key role in understanding model stability, robustness, and generalization. Constraining the spectral norm of neural network layers has been shown to improve training stability (Miyato et al., 2018), enhance generalization (Bartlett et al., 2017), and provide certified robustness (Tsuzuku et al., 2018; Virmaux and Scaman, 2018).

Despite its importance, efficiently computing the spectral norm remains a challenge, particularly for deep neural networks where layers may be high-dimensional or

structured, such as convolutional layers. Existing approaches, such as power iteration (PI), provide estimates but lack strict guarantees on approximation quality, slow in convergence (linear), and can be sensitive to initialization.SVD methods offer exact computation but are computationally expensive on GPUs. This limits their effectiveness in applications requiring certified bounds, such as adversarial robustness and constrained optimization.

In this chapter, we introduce a novel method for spectral norm estimation that provides a strict upper bound at every iteration, addressing the limitations of prior techniques while achieving super geometric convergence, contrary to PI it is not a matrix-free method. Our approach extends beyond fully connected layers to convolutional layers, including both circular and zero-padding cases. While the spectral norm is traditionally defined via the explicit weight matrix, we also provide a matrix-free formulation that operates directly on the convolutional filter, even when the corresponding matrix is not explicitly constructed.

Unlike power iteration, our method is deterministic, ensuring stability and reproducibility, and integrates seamlessly into neural network training pipelines for spectral regularization. Indeed, we provide an explicit gradient formulation for backpropagation, enabling spectral regularization during training. Additionally, we leverage GPU acceleration to maintain computational efficiency, making our approach scalable to large-scale models. Code is available at this link `https://github.com/blaisedelattre/lip4conv`.

3.1 Spectral norm estimation for matrices

The Frobenius norm of $\mathbf{W}$ is represented as $\|\mathbf{W}\|_{\mathrm{F}}$. A matrix norm $\|\cdot\|$ is said to be *consistent* if it satisfies the submultiplicative property: for any matrices $\mathbf{A}$ and $\mathbf{B}$,

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|.$$

The Kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted as $\mathbf{A} \otimes \mathbf{B}$. The conjugate transpose of a matrix $\mathbf{W}$ is denoted by $\mathbf{W}^*$.

The spectral norm of a matrix $\mathbf{W} \in \mathbb{C}^{p \times q}$, which represents a linear operator, is defined as its largest singular value. Formally, the spectral norm $\|\mathbf{W}\|_2$ is given by:

$$\|\mathbf{W}\|_2 = \sup_{\|\mathbf{x}\|_2 = 1} \|\mathbf{W}\mathbf{x}\|_2,$$

where $\mathbf{x}$ is any vector in the input space. This norm quantifies the maximum amplification of a unit-norm vector under the transformation induced by $\mathbf{W}$. Consequently, computing the spectral norm requires evaluating the singular values of the matrix.

Definition 3.1.1 (Singular value decomposition and singular values)**.**
*For a matrix $\mathbf{W} \in \mathbb{C}^{p\times q}$, the Singular Value Decomposition (*SVD*) is a factorization of the form:*

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*,$$

where $\mathbf{U} \in \mathbb{C}^{p\times p}$ and $\mathbf{V} \in \mathbb{C}^{q\times q}$ are unitary matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{p\times q}$ is a diagonal matrix with non-negative entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(p,q)} \geq 0$, called the singular values of $\mathbf{W}$.

The spectral norm $\|\mathbf{W}\|_2$ equals the largest singular value $\sigma_1(\mathbf{W})$. Among the most widely used methods for computing the spectral norm are the Singular Value Decomposition (SVD) via QR method and Power Iteration (PI).

Singular value decomposition provides an exact computation of the spectral norm by directly extracting the largest singular value (Golub et al., 1996). However, its computational complexity is $O(pq^2)$ for a matrix of size $p \times q$, making it expensive for large matrices. Iterative methods, such as the Lanczos (Lanczos, 1950) or Arnoldi (Arnoldi, 1951) methods, offer more efficient alternatives but remain computationally intensive. The QR method with shift is another iterative approach used to compute the SVD (Golub et al., 1996). This method transforms the matrix into a simpler form (e.g., upper triangular) and uses shifts to accelerate convergence. While QR methods are numerically stable and converge faster with shifts, they are not implemented on GPUs in frameworks like PyTorch, which instead rely on the divide-and-conquer algorithm for SVD computation. This divide-and-conquer approach, although accurate, can be slower for large matrices. Moreover, QR methods are known to be ill-conditioned for differentiation, making them unsuitable for applications requiring backpropagation through SVD (Wang et al., 2022). Ill-conditioning arises from the sensitivity of singular values and singular vectors to small perturbations in the input matrix, causing instability during gradient computation.

That is why we focus on iterative methods, which are more suitable for backpropagation and can be implemented on GPUs for efficient parallel computation.

Power iteration (PI) is a classical solution that serves as the basis for many others, and it is described in Algorithm 2. The method iteratively applies the matrix $\mathbf{W}$ to a random vector $\mathbf{u}$ and normalizes the result to obtain the largest singular value $\sigma_1(\mathbf{W})$. The process is repeated for a fixed number of iterations N_{iter} until convergence. The convergence rate of the PI is linear, with a rate of σ_1/σ_2, where σ_1

and σ_2 are the largest and second-largest singular values of $\mathbf{W}$, respectively. The method is not fully deterministic, as it starts from a randomly generated vector, and intermediate iterations are not guaranteed to be a strict upper bound on the spectral norm. PI has been used to compute the dominant principal component efficiently for high-dimensional data (Halko et al., 2011), Google's PageRank algorithm relies on power iteration to compute the leading eigenvector of the hyperlink matrix (Page et al., 1999), used to extract dominant spectral features of adjacency matrices in graph learning and spectral clustering (Von Luxburg, 2007).

Algorithm 2 Power_iteration($\mathbf{W}, N_{\text{iter}}$)

Inputs: matrix $\mathbf{W}$, number of iterations N_{iter}
Initialization: draw a random vector $\mathbf{u}$
for $i = 1$ to N_{iter}
    $\mathbf{v} \leftarrow \mathbf{W}\mathbf{u}/\|\mathbf{W}\mathbf{u}\|_2$
    $\mathbf{u} \leftarrow \mathbf{W}^*\mathbf{v}/\|\mathbf{W}^*\mathbf{v}\|_2$
$\sigma_1 \leftarrow (\mathbf{W}\mathbf{u})^*\mathbf{v}$
return σ_1

To quantify the rate of convergence of the Power Iteration, we recall the notions of Q-linear and R-linear convergence.

Definition 3.1.2 (Q-linear and R-linear convergence (Nocedal and Wright, 2006))**.** *Suppose the sequence* (b_k) *converges to a limit* L*. The sequence is said to converge* Q-linearly *if, for some* $\mu \in (0, 1)$*:*

$$\lim_{k\to\infty} \frac{|b_{k+1} - L|}{|b_k - L|} = \mu.$$

Moreover, the sequence is said to converge R-linearly *if there exists another sequence* (ϵ_k) *such that:*

$$|b_k - L| \leq \epsilon_k, \quad \forall k,$$

and (ϵ_k) *converges Q-linearly to* 0*, i.e.,*

$$\lim_{k\to\infty} \frac{\epsilon_{k+1}}{\epsilon_k} = \mu.$$

For PI the convergence is Q-linear with a convergence rate of $\mu = \sigma_1/\sigma_2$. This ratio can be close to 1 and other variations of PI improve it (Lanczos, 1950; Arnoldi, 1951; Kublanovskaya, 1962) and more recently in Lehoucq et al. (1996). However, these methods suffer from important drawbacks. First, many iterations are required to ensure convergence, up to zero numerical precision, which can be computationally

expensive. Second, the method is not fully deterministic, as it starts from a randomly generated vector. Finally, intermediate iterations are not guaranteed to be a strict upper bound on the spectral norm.

Lower bound by Power Iteration Here we explain why PI provides a lower bound on the spectral norm for square matrices (without loss of generality). Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ and $\|\mathbf{W}\|_2 = \max_{\|u\|_2=1} \|\mathbf{W}u\|_2$. Power iteration generates

$$\mathbf{u}_{k+1} = \frac{\mathbf{W}\mathbf{u}_k}{\|\mathbf{W}\mathbf{u}_k\|_2}, \qquad \|\mathbf{u}_k\|_2 = 1.$$

At each step,

$$\|\mathbf{W}\mathbf{u}_k\|_2 \leq \|\mathbf{W}\|_2,$$

so $\|\mathbf{W}\mathbf{u}_k\|_2$ is a lower bound on the spectral norm. Moreover, since $\mathbf{u}_k \to \mathbf{v}_1$ (dominant eigenvector),

$$\lim_{k \to \infty} \|\mathbf{W}\mathbf{u}_k\|_2 = \|\mathbf{W}\|_2.$$

3.2 Gram iteration

To remedy those issues, we present *Gram iteration* (GI), a novel method for estimating an upper bound on the spectral norm of a matrix optimized for GPU acceleration. It is based on repetitive Gram matrix computation.

Definition 3.2.1 (Gram matrix)**.**
Given a matrix $\mathbf{W} \in \mathbb{C}^{p \times q}$, the Gram matrix associated with $\mathbf{W}$ is defined as

$$\mathbf{W}^\star \mathbf{W} \in \mathbb{C}^{q \times q},$$

where $\mathbf{W}^\star$ denotes the transpose (or Hermitian transpose in the complex case) of $\mathbf{W}$.

Detailed proofs can be found in Appendix A. We refer to the iterative method outlined in Theorem 3.2.4 as Gram iteration due to its structure: at each step, the matrix $\mathbf{W}^{(t+1)}$ is the Gram matrix of $\mathbf{W}^{(t)}$, and $\sigma_1(\mathbf{W}^{(t+1)}) = \sigma_1(\mathbf{W}^{(t)})^2$

$$\begin{cases} \mathbf{W}^{(1)} = \mathbf{W}, \\ \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)\star} \mathbf{W}^{(t)}. \end{cases} \tag{3.1}$$

$\mathbf{W}^{(t)}$ is called the t-th iterate of the Gram iteration, Gram iteration algorithm is presented in Algorithm 3 for any matrix norm $\|.\|$. Note that this algorithm does not include rescaling to avoid overflow, which is presented in Algorithm 4. At the

Algorithm 3 Gram_iteration_naive($\mathbf{W}, N_{\text{iter}}$)

1: **Inputs:** matrix $\mathbf{W}$, number of iterations N_{iter}
2: **for** $i = 1$ to N_{iter}
3: $\quad \mathbf{W} \leftarrow \mathbf{W}^*\mathbf{W}$
4: $\sigma_1 \leftarrow \|\mathbf{W}\|^{2^{-N_{\text{iter}}}}$
5: **return** $\sigma_1, \mathbf{W}$

end of the Gram iteration, the dominant singular vectors of $\mathbf{W}$ can be estimated. The final matrix $\mathbf{W}$, obtained through repeated applications of $\mathbf{W}^*\mathbf{W}$, acts as a projection onto the subspace of the largest singular value. As iterations increase, $\mathbf{W}$ converges to a rank-one projector aligned with the dominant singular vector. Thus, any nonzero column $\mathbf{W}_{:,i}$ is proportional to the largest right-singular vector $\mathbf{u}$, which can be estimated as:

$$\mathbf{u} \leftarrow \frac{\mathbf{W}_{:,i}}{\|\mathbf{W}_{:,i}\|_2}. \tag{3.2}$$

where i is any column index such that $\mathbf{W}_{:,i} \neq 0$.

Largest eigenpair by squaring iteration Let $\mathbf{W}$ be a square complex matrix, $\mathbf{W} \in \mathbb{C}^{p\times p}$. We aim to compute its largest eigenpair (λ_1, u_1), where λ_1 is the largest eigenvalue and u_1 is the corresponding eigenvector. Algorithm 3 can be modified to extract the dominant eigenpair by replacing Step 3 with squaring the matrix $\mathbf{W}$ at each iteration:

$$\mathbf{W} \leftarrow \mathbf{W}\mathbf{W} \tag{3.3}$$

Given the eigenvalue decomposition $\mathbf{W} = Q\Lambda Q^{-1}$, the t-th iterate satisfies

$$\mathbf{W}^{(t)} = \mathbf{W}^{2^{t-1}} = Q\Lambda^{2^{t-1}}Q^{-1} \tag{3.4}$$

As $t \to \infty$, the largest eigenvalue λ_1 dominates (if $\lambda_1 > \lambda_2$), and $\mathbf{W}^{(t)}$ converges to a rank-one approximation aligned with the dominant eigenvector. Since $\mathbf{W}^{(t)}$ approximates a rank-one projector, any nonzero column $\mathbf{W}_{:,i}$ is proportional to $\mathbf{u}_1$, allowing its estimation as

$$\mathbf{u}_1 \leftarrow \frac{\mathbf{W}_{:,i}}{\|\mathbf{W}_{:,i}\|_2}, \quad \text{where } i \text{ is any nonzero column index} \tag{3.5}$$

Definition 3.2.2 (Gelfand's formula for spectral radius (Horn and Johnson, 2012))**.** *Let $\mathbf{W}$ be a square matrix over $\mathbb{C}^{p\times q}$ and $\|\cdot\|$ any matrix norm. The spectral radius of $\mathbf{W}$, denoted $\rho(\mathbf{W})$, is given by the limit*

$$\rho(\mathbf{W}) = \lim_{t\to\infty} \|\mathbf{W}^t\|^{1/t}.$$

This identity, known as Gelfand's formula, always holds in finite-dimensional spaces, regardless of the choice of norm. Moreover, if $\|\cdot\|$ is sub-multiplicative, then the sequence $\|\mathbf{W}^t\|^{1/t}$ satisfies

$$\rho(\mathbf{W}) \leq \|\mathbf{W}^t\|^{1/t}, \qquad \text{for all } t \in \mathbb{N},$$

and thus converges to $\rho(\mathbf{W})$ from above.

Gram iteration uses Gelfand's formula 3.2.2 on Gram matrix, as $\|\mathbf{W}\|_2^2 = \rho(\mathbf{W}^\star\mathbf{W})$, and applies matrix squaring to generate a sequence of iterates that converge to the spectral norm.

Convergence analysis and upper bound guarantee To quantify the rate of convergence of the GI, we recall the notions of Q-quadratic R-quadratic and super geometric convergence. Linear convergence reduces the error proportionally, while quadratic convergence reduces the error proportionally to its square, making quadratic much faster as it approaches the limit. Super geometric convergence, on the other hand, implies that the error decays faster than any linear converging sequence, leading to even more rapid convergence but slower than quadratic.

Definition 3.2.3 (Q-quadratic, R-quadratic, and supergeometric convergence)**.** *Suppose the sequence (b_k) converges to a limit L.*

- *The sequence converges* Q-quadratically *if there exists a constant $\mu > 0$ such that*

$$\lim_{k\to\infty} \frac{|b_{k+1} - L|}{|b_k - L|^2} = \mu\,.$$

- *The sequence converges* R-quadratically *if there exists a nonnegative sequence (ϵ_k) with $|b_k - L| \leq \epsilon_k$ for all k, and (ϵ_k) converges Q-quadratically to 0.*

- *More generally, for $p > 1$, we say (b_k) converges* R-superlinearly of order p *if there exists $K > 0$ such that*

$$|b_{k+1} - L| \;\leq\; K\,|b_k - L|^p \qquad \text{for all } k \text{ sufficiently large.}$$

  Quadratic convergence corresponds to $p = 2$.

- *Following Friedland (Friedland, 1991), the sequence is said to converge* supergeometrically of order p with rate $K \in (0,1)$ *if for every $\varepsilon > 0$ there exist constants $C > 0$ and N such that*

$$|b_k - L| \;\leq\; C\,(K + \varepsilon)^{p^k} \qquad \forall k \geq N.$$

This means the error decays faster than any geometric sequence c^k, and with a double-exponential speed in p^k.

These results are formalized in the following theorem.

Theorem 3.2.4 (Gram iteration: convergence, upper bound, and supergeometric rate (Frobenius))**.**
Let $\|\cdot\|$ be a submultiplicative matrix norm and let $\mathbf{W} \in \mathbb{C}^{p\times q}$. Define $\mathbf{W}^{(1)} = \mathbf{W}$ and $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)*}\mathbf{W}^{(t)}$, and set

$$s_t := \|\mathbf{W}^{(t)}\|^{2^{1-t}}, \qquad t \geq 1.$$

Then, for all $t \geq 2$:

1. *(*Convergence*)* $s_t \to \sigma_1(\mathbf{W})$ *as* $t \to \infty$.

2. *(*Upper bound*) For any matrix norm,* $s_t \geq \sigma_1(\mathbf{W})$.

Moreover, the following rate refinements *hold.*

*(a) (*General norm*) In general, one has the R-linear bound*

$$0 \leq s_t - \sigma_1(\mathbf{W}) \leq C\,2^{-t} \quad \textit{for all large } t,$$

for some constant $C > 0$ depending on the norm and the dimension.

*(b) (*Frobenius readout: supergeometric and "order $\to 2$"*) If the final readout is the Frobenius norm, then the error $e_t := s_t - \sigma_1(\mathbf{W})$ is* supergeometric *of order 2 with rate $\mu = \sigma_2(\mathbf{W})/\sigma_1(\mathbf{W}) \in (0,1)$: for every $\varepsilon > 0$ there exist $C_\varepsilon > 0$ and T such that*

$$0 \leq e_t \leq C_\varepsilon\,(\mu+\varepsilon)^{2^t} \qquad \forall t \geq T.$$

In particular, the local (log–log) order

$$p_t := \frac{\log e_{t+1}}{\log e_t}$$

satisfies $p_t = 2 - \Theta(t/2^t)$; hence $p_t \uparrow 2$ and the convergence is superlinear *and* arbitrarily close to quadratic *(as t grows), but* not *R-quadratic in general.*

See proof in Appendix A.0.3.

The first part of Theorem 3.2.4 was already noted in Friedland (1991), but only for square matrices and without any practical implementation or experimental validation (e.g. on GPUs).

It is also important to clarify the type of convergence obtained. In classical numerical analysis, *quadratic convergence* means that $e_{t+1} \leq \mu e_t^2$ for some constant $\mu > 0$, which is equivalent to the local order $p_t = \log(e_{t+1})/\log(e_t)$ tending to 2. In the Gram iteration with Frobenius readout we established instead

$$e_t = \Theta\left(\frac{q^{2^t}}{2^t}\right), \qquad q = \tfrac{\sigma_2}{\sigma_1} < 1,$$

which is *supergeometric* in the sense of Friedland (double–exponential decay). The local order p_t still tends to 2, so the convergence is superlinear and appears "almost quadratic," but it is *not* R–quadratic since $e_{t+1}/e_t^2 \sim 2^t \to \infty$.

In practice, this provides a sharp contrast with the classical *power iteration*, which converges only linearly with rate σ_2/σ_1. Gram iteration thus achieves convergence that is faster than any linear rate and empirically nearly indistinguishable from quadratic, but without the uniform guarantee of true quadratic convergence.

Algorithm and implementation We present in the following algorithm 4 a practical implementation of the Gram iteration method with Frobenius norm to have an upper bound on spectral norm at any iteration. To avoid overflow, matrix $\mathbf{W}$ is rescaled at each iteration, and scaling factors are accumulated in variable r to unscale the result at the end of the method. Unscaling is crucial as it is required to remain a strict upper bound on the spectral norm at each iteration of the method. One can note that GI provides a proper norm at each iteration, which is a deterministic upper bound on the spectral norm, and which converges super geometrically. However, GI is not a matrix-free method, contrary to power iteration (PI) and its variants.

Algorithm 4 Gram_iteration($\mathbf{W}, N_{\text{iter}}$)

Inputs: matrix $\mathbf{W}$, number of iterations N_{iter}
$r \leftarrow 0$ // Initialize rescaling
for $t = 1$ to N_{iter}
    $r \leftarrow 2(r + \log\|\mathbf{W}\|_{\text{F}})$ // Accumulate rescaling
    $\mathbf{W} \leftarrow \mathbf{W}/\|\mathbf{W}\|_{\text{F}}$ // Rescale to avoid overflow
    $\mathbf{W} \leftarrow \mathbf{W}^*\mathbf{W}$ // Gram iteration
$\sigma_1 \leftarrow \|\mathbf{W}\|^{2^{-N_{\text{iter}}}} \exp\left(2^{-N_{\text{iter}}} r\right)$
return σ_1

Explicit gradient computation In the context of differentiating through the spectral norm of a layer, ensuring the differentiability of the upper bound is necessary. For the

Frobenius norm, an explicit gradient formulation can be derived. Interestingly, the bound sequence $\left(\|\mathbf{W}^{(t)}\|_{\mathrm{F}}^{2^{1-t}}\right)$ aligns with the Schatten 2^{1-t}-norm of $\mathbf{W}$, providing a useful connection between norm approximations and their gradients.

Proposition 3.2.5 (Gradient of the Gram iteration bound)**.**
For the Frobenius norm, the bound sequence $(\|\mathbf{W}^{(t)}\|_F^{2^{1-t}})$ *is differentiable with respect to* $\mathbf{W}$*, and its gradient is given by:*

$$\frac{\partial(\|\mathbf{W}^{(t)}\|_{\mathrm{F}}^{2^{1-t}})}{\partial \mathbf{W}} = \frac{\mathbf{W}(\mathbf{W}^*\mathbf{W})^{2^{t-1}-1}}{\|\mathbf{W}^{(t)}\|_{\mathrm{F}}^{2(1-2^{-t})}} \,. \tag{3.6}$$

This proposition is essential as it enables the direct computation of a gradient for optimization, allowing efficient backpropagation without the overhead of maintaining a large computational graph. However, computing this gradient involves matrix exponentiation, which can be numerically unstable. To mitigate this issue, a practical approach is to rescale by an estimate of the spectral norm, computed during the forward pass.

While our focus so far has been on linear layers, structured layers like convolutional layers are equally critical due to their widespread use in deep learning models such as convolution neural networks, particularly in computer vision (LeCun et al., 1998; Krizhevsky et al., 2012; Tan and Le, 2021). Deriving spectral norm bounds for these structured layers is essential to extend the robustness and generalization guarantees of spectral regularization techniques to more complex architectures (Tsuzuku et al., 2018; Sedghi et al., 2019). In the following, we address this challenge and provide bounds adapted to convolutional layers.

3.3 Related works on convolutional layers

Convolutional layers are the main building block of Convolutional Neural Networks (CNNs) (LeCun et al., 1998), which have become pivotal in computer vision and achieve state-of-the-art performance (Krizhevsky et al., 2012; Tan and Le, 2021). The essence of CNNs lies in a cascade of convolutional layers, crucial for extracting hierarchical features. Convolutional architecture gradually reduces the spatial dimensions of input while increasing the semantic dimension and maintaining the separability of classes (Mallat, 2016). CNNs are both sparse and structured: indeed the weight-sharing topology of such networks encodes inductive biases, like locality and translation-equivariance. From these biases arises a very powerful architecture for computer vision, in terms of speed, parameter efficiency, and performance.

Beyond visual tasks, convolutional layers contribute to applications like natural language and speech processing (Conneau et al., 2017; Baevski et al., 2020; Gu et al., 2022; Li et al., 2022) (mainly with one-dimensional convolution), and to graph data (Zhang et al., 2019b) for applications in biology for instance.

Despite their long-standing use, CNNs are still the focus of lively research on many theoretical and empirical aspects. For instance, the relationship between the dataset structure and CNN architectures is explored using spectral theory (Pinson et al., 2023). Examining CNNs from a different perspective, some approaches investigate their properties through the Hessian rank of their matrix representation (Singh et al., 2023). Padding in convolutional layers is the subject of various studies, analyzing its impact on frequency (Tang et al., 2023), adversarial robustness (Gavrikov and Keuper, 2023), and encoded position information (Islam et al., 2019; Semih Kayhan and Van Gemert, 2020). The effects of stride on convolutional and pooling layers, particularly addressing aliasing issues, are considered and corrected within the framework of sampling theory (Zhang, 2019). All these contributions showed that CNNs' layers and their design can still be improved, thanks to theoretical exploration.

3.4 Related works on the spectral norm of a convolutional layer

Let $\mathbf{X} \in \mathbb{R}^{c_{\text{in}} \times n \times n}$ be an input image, $\mathbf{x} \in \mathbb{R}^{c_{\text{in}} n^2}$ its vectorized form, and $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k \times k}$ a convolution filter, and k the kernel size. Throughout this section, unless stated otherwise, K is assumed to be padded to match the input size, i.e., $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times n \times n}$, using both zero padding and circular padding. Applying maximum non-trivial zero padding extends the input $\mathbf{X} \in \mathbb{R}^{c_{\text{in}} \times n \times n}$ to $\mathbb{R}^{c_{\text{in}} \times (n+k-1) \times (n+k-1)}$.

For $1 \leq j \leq c_{\text{out}}$ and $1 \leq u, v \leq n$, the convolution output $\mathbf{Y} = K \star \mathbf{X}$ is given by:

$$\mathbf{Y}_{j,u,v} = \sum_{i=1}^{c_{\text{in}}} \sum_{k_1=1}^{n} \sum_{k_2=1}^{n} \mathbf{X}_{i,u+k_1,v+k_2} K_{j,i,k_1,k_2}. \tag{3.7}$$

In practice, the Im2Col (Chellapilla et al., 2006) representation is often used to express convolution as a matrix multiplication by patching the input $\mathbf{X}$ and vectorizing the filter K, significantly improving computational efficiency and memory access patterns. However, this approach is not suited for spectral norm computation. Instead we represent the convolution operator $\star$ explicitly as a matrix-vector product,

Tab. 3.1.: Qualitative summary of estimations of Lipschitz constant of convolutional layers. Precision and speed qualifications come from experimental results. "deter" abbreviates "deterministic". Ryu et al. (2019) uses PI adapted to convolution to compute an estimate arbitrarily precise in a zero padding (a special case of constant padding) setting but suffers from slow convergence. Araujo et al. (2021) uses the Toeplitz matrix theory to devise a fast bound despite precision. Other approaches suppose circular padding for convolution which is a reasonable (comparable performance) and useful assumption to reduce computation complexity thanks to the Fourier transform. Sedghi et al. (2019) provides exact computation of singular values, however, the method is very slow and does not scale. Singla and Feizi (2021a) proposes a faster using PI method to the detriment of precision. Methods using PI are not guaranteed to produce an upper bound on convolution spectral norm regarding constant or circular padding. Our method gathers the best of both worlds: being fast, precise, deterministic, well differentiable, and with a strict upper bound on the spectral norm.

Methods	Precision	Speed	Padding	Upper Bound	Algorithm	Convergence
Farnia et al.; Ryu et al.	++	-	zero	✗	PI: iterative, non-deter	linear
Bibi et al.; Sedghi et al.	++	- -	circular	exact	SVD: deter	-
Yi; Singla and Feizi	+	+	circular	✗	PI: iterative, non-deter	linear
Araujo et al.	-	++	zero	✓	close-form, deter	-
Ours	++	++	circular	✓	GI: iterative, deter	super geometric (near quadratic)

where $\mathbf{W} \in \mathbb{R}^{c_{\text{out}}n^2 \times c_{\text{in}}n^2}$ encodes the filter K. This formulation explicitly illustrates the weight-sharing property of convolution (Jain, 1989):

$$\mathbf{y} = \text{vec}(K \star \mathbf{X}) = \mathbf{W}\mathbf{x}, \tag{3.8}$$

although it is highly inefficient for memory storage due to the large size of $\mathbf{W}$.

Estimating the spectral norm of convolutional layers is challenging. The matrix associated with a convolution has size $n^2 c_{\text{in}} \times n^2 c_{\text{out}}$ for an input $\mathbf{x} \in \mathbb{R}^{c_{\text{in}} \times n \times n}$, making direct methods such as the SVD computationally infeasible. The difficulty is compounded for zero-padded convolutions, where Fourier diagonalization no longer applies and no closed-form formula is available. A first line of work develops *matrix-free* estimators based on power iteration (PI), which only require applying the convolution as a matrix–vector product. Ryu et al. (2019) introduced a convolution-adapted PI whose runtime scales with the convolution itself and can be efficiently accelerated on GPUs, but the linear convergence of PI limits accuracy. Singla and Feizi (2021a) proposed a related method that computes an upper bound by taking the minimum spectral norm across four transformed matrices; while efficient, this bound can be loose, especially for zero-padded convolutions (Yi, 2020). A second line of work exploits the structural properties of convolutions. Sedghi et al. (2019) represented convolutional layers as doubly-block circulant matrices, which can be block-diagonalized with the Fourier transform; applying SVD on the resulting blocks yields exact spectral norms under certain assumptions, but the cost remains prohibitive for large filters. Bibi et al. (2019) similarly proposed exact but computationally expensive formulations. More recent approaches aim at scalable

upper bounds. Araujo et al. (2021) derived efficient bounds that are exact for certain filter shapes (e.g. $c_{\text{out}} \times 1 \times k \times k$ or $1 \times c_{\text{in}} \times k \times k$) but do not generalize to all cases. Yi (2020) showed that such bounds remain valid even with zero padding. However, methods relying on PI (Ryu et al., 2019; Singla and Feizi, 2021a) do not guarantee upper bounds, which restricts their applicability for robustness certification.

While recent methods have improved scalability, they often sacrifice precision or lack formal guarantees, see Table 3.1 for a summary. Estimating the spectral norm of convolutional layers efficiently remains an open challenge, balancing precision and computational cost. Our work addresses this gap by leveraging the *Gram iteration* approach, which provides a scalable and certified method for bounding the spectral norm of convolutional layers.

3.4.1 Zero-Padded convolutions

In typical CNNs, convolutional layers are often applied with zero padding to ensure the output $\mathbf{Y}$ has the same spatial dimensions as the input, assuming a stride of 1. The shared weights of convolutional filters result in a Toeplitz structure (Jain, 1989). This can be expressed as:

$$\text{vec}(\mathbf{Y}) = \mathbf{W}_{\text{toep}}\mathbf{x}, \tag{3.9}$$

where $\mathbf{W}_{\text{toep}} \in \mathbb{R}^{c_{\text{out}}n^2 \times c_{\text{in}}n^2}$ is a $c_{\text{out}} \times c_{\text{in}}$ block matrix. Each block $\mathbf{W}_{\text{toep}_{j,i}}$ is a $n^2 \times n^2$ banded doubly Toeplitz matrix formed from the kernel $K_{j,i}$. A detailed representation of the Toeplitz matrix is provided in Appendix A.0.1. As noted in (Yi, 2020), $\mathbf{W}_{\text{toep}}$ is a multi-level block Toeplitz matrix with $c_{\text{out}} \times c_{\text{in}}$ entries. The Lipschitz constant of the convolutional layer is determined by the maximum singular value of $\mathbf{W}_{\text{toep}}$, $\sigma_1(\mathbf{W}_{\text{toep}})$. Direct computation of $\sigma_1(\mathbf{W}_{\text{toep}})$ is computationally expensive; hence, Fourier transform techniques are often used to simplify the analysis of Toeplitz matrices.

3.4.2 Circular-Padded convolutions

Let us consider first the case where $c_{\text{in}} = c_{\text{out}} = 1$ to simplify the analysis. When circular padding is applied, the convolution matrix no longer has a Toeplitz structure but instead becomes doubly-block circulant. Specifically, $\mathbf{W}_{\text{circ}}$ is a doubly-block circulant matrix, where each row block is a circular shift of the previous one, and each block itself is circulant (Jain, 1989). Denoting $\text{circ}(\cdot)$ as the operator that generates a circulant matrix from a vector, the structure of $\mathbf{W}_{\text{circ}}$ is entirely determined by the

convolutional kernel K. A detailed representation can be found in Appendix A.0.2. This matrix can be expressed as:

$$\mathbf{W}_{\text{circ}} = \begin{pmatrix} \text{circ}(K_1) & \text{circ}(K_2) & \cdots & \text{circ}(K_n) \\ \text{circ}(K_n) & \text{circ}(K_1) & \ddots & \vdots \\ \vdots & \ddots & \text{circ}(K_1) & \text{circ}(K_2) \\ \text{circ}(K_2) & \cdots & \text{circ}(K_n) & \text{circ}(K_1) \end{pmatrix}. \tag{3.10}$$

For simplicity, we denote $\mathbf{W}_{\text{circ}} = \text{blkcirc}(K_1, \ldots, K_n)$. The choice of padding significantly impacts the convolution's properties. With zero padding, the matrix is doubly-block Toeplitz, and properties such as Fourier diagonalization do not apply. In contrast, the doubly-block circulant structure introduced by circular padding enables efficient computation using Fourier transforms. $\mathbf{W}_{\text{circ}}$, it can be diagonalized using the discrete Fourier transform (DFT).

Definition 3.4.1 (Discrete Fourier Transform Matrix).
Let $\omega := e^{-2\pi \mathbf{i}/n}$. The DFT matrix $\mathbf{U} \in \mathbb{C}^{n \times n}$ is

$$\mathbf{U}_{u,v} = \omega^{(u-1)(v-1)}, \qquad 1 \le u, v \le n,$$

with inverse $\mathbf{U}^{-1} = \frac{1}{n}\mathbf{U}^*$.

For a 2D input $\mathbf{X}$, the 2D discrete Fourier transform can be expressed as:

$$\text{vec}(\mathbf{U}\mathbf{X}\mathbf{U}^\top) = (\mathbf{U} \otimes \mathbf{U})\mathbf{x}, \tag{3.11}$$

For brevity, we denote $\mathbf{F} = \mathbf{U} \otimes \mathbf{U}$.

Theorem 3.4.2 (Diagonalization of Circulant Matrices (Jain, 1989)).
Let $K \in \mathbb{R}^{n \times n}$ be a convolutional kernel and $\mathbf{W}_{\text{circ}} \in \mathbb{R}^{n^2 \times n^2}$ be the doubly-block circulant matrix such that $\mathbf{W}_{\text{circ}} = \text{blkcirc}(K_1, \ldots, K_n)$. Then, $\mathbf{W}_{\text{circ}}$ can be diagonalized as:

$$\mathbf{W}_{\text{circ}} = \mathbf{F}\text{diag}(\boldsymbol{\lambda})\mathbf{F}^{-1}, \tag{3.12}$$

where $\boldsymbol{\lambda} = \mathbf{F}\text{vec}(K)$ are the eigenvalues of $\mathbf{W}_{\text{circ}}$.

In the context of deep learning, convolutions are applied with multiple channels: c_{out} convolutions are applied on input with c_{in} channels. Therefore, the matrix associated with the convolutional filter $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times n \times n}$ becomes a block matrix, where each block is a doubly-block circulant matrix: $\mathbf{W}_{\text{circ}} = (\mathbf{W}_{\text{circ}i,j})$, for $1 \le i \le c_{\text{out}}$ and

$1 \le j \le c_{\text{in}}$, with $\mathbf{W}_{\text{circ}i,j} = \text{blkcirc}(K_{i,j,1}, \ldots, K_{i,j,n})$. As also studied by Sedghi et al. (2019) and Yi (2020), this type of block doubly-block circulant matrix can be *block* diagonalized as follows:

Theorem 3.4.3 (Block diagonalization of multi-channel circular convolution (Trockman and Kolter, 2021, Cor. A.1.1))**.**
Let $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times n \times n}$ be a circular convolutional filter and let $\mathbf{W}_{\text{circ}} \in \mathbb{R}^{c_{\text{out}} n^2 \times c_{\text{in}} n^2}$ denote the corresponding convolution matrix. Let $\mathbf{P}_{\text{out}} \in \mathbb{R}^{c_{\text{out}} n^2 \times c_{\text{out}} n^2}$ and $\mathbf{P}_{\text{in}} \in \mathbb{R}^{c_{\text{in}} n^2 \times c_{\text{in}} n^2}$ be permutation matrices, and let $\mathbf{F} = \mathbf{U} \otimes \mathbf{U}$ be the 2D DFT matrix. Then $\mathbf{W}_{\text{circ}}$ admits the block diagonalization

$$\mathbf{W}_{\text{circ}} = (\mathbf{I}_{c_{\text{out}}} \otimes \mathbf{F})\, \mathbf{P}_{\text{out}}\, \mathbf{D}\, \mathbf{P}_{\text{in}}^{\top}\, (\mathbf{I}_{c_{\text{in}}} \otimes \mathbf{F}^{-1}), \tag{3.13}$$

where $\mathbf{I}_{c_{\text{out}}}$ and $\mathbf{I}_{c_{\text{in}}}$ are identity matrices of size c_{out} and c_{in}, respectively, and $\mathbf{D}$ is block diagonal with n^2 diagonal blocks of size $c_{\text{out}} \times c_{\text{in}}$.

Each block of $\mathbf{D}$ corresponds to one spatial frequency and collects the Fourier coefficients of the filters across channels. Explicitly,

$$\mathbf{D} = \mathbf{P}_{\text{out}}^{\top} \begin{pmatrix} \mathbf{F}\,\text{vec}(K_{1,1}) & \cdots & \mathbf{F}\,\text{vec}(K_{1,c_{\text{in}}}) \\ \vdots & & \vdots \\ \mathbf{F}\,\text{vec}(K_{c_{\text{out}},1}) & \cdots & \mathbf{F}\,\text{vec}(K_{c_{\text{out}},c_{\text{in}}}) \end{pmatrix} \mathbf{P}_{\text{in}}.$$

Explicit permutations $\mathbf{P}_{\text{out}}$ and $\mathbf{P}_{\text{in}}$ Let $\mathbf{e}_k$ denote the k-th canonical basis vector. We index outputs by (i, f) with $1 \le i \le c_{\text{out}}$ (channel) and $1 \le f \le n^2$ (frequency), so that the row index is $(i-1)n^2 + f$. Similarly, inputs are indexed by (j, f) with $1 \le j \le c_{\text{in}}$, giving column index $(j-1)n^2 + f$. This corresponds to the usual *channel-major* ordering. The permutation matrices $\mathbf{P}_{\text{out}} \in \mathbb{R}^{c_{\text{out}} n^2 \times c_{\text{out}} n^2}$ and $\mathbf{P}_{\text{in}} \in \mathbb{R}^{c_{\text{in}} n^2 \times c_{\text{in}} n^2}$ reorder these indices as

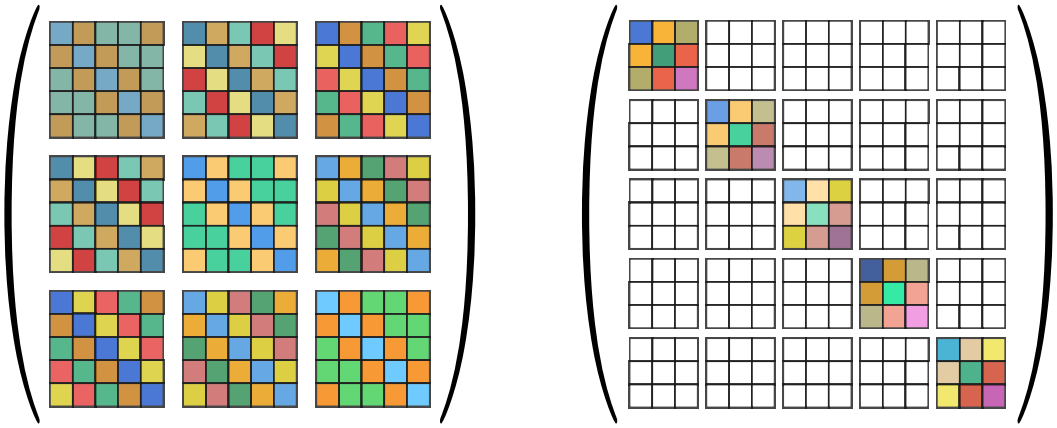
$$(\mathbf{P}_{\text{out}})_{(f-1)c_{\text{out}}+i,\,(i-1)n^2+f} = 1, \qquad (\mathbf{P}_{\text{in}})_{(f-1)c_{\text{in}}+j,\,(j-1)n^2+f} = 1,$$

with all other entries 0. Equivalently,

$$\mathbf{P}_{\text{out}}\mathbf{e}_{(i-1)n^2+f} = \mathbf{e}_{(f-1)c_{\text{out}}+i}, \qquad \mathbf{P}_{\text{in}}\mathbf{e}_{(j-1)n^2+f} = \mathbf{e}_{(f-1)c_{\text{in}}+j}.$$

Thus $\mathbf{P}_{\text{out}}$ and $\mathbf{P}_{\text{in}}$ convert vectors from *channel-major* to *frequency-major* order, with their transposes performing the inverse.

In the block-diagonalization of convolutional layers, $\mathbf{P}_{\text{out}}$ and $\mathbf{P}_{\text{in}}$ simply reshuffle the matrix $\mathbf{D}$ so that it becomes block diagonal. This corresponds to adopting an alternative vectorization of the layer input (Sedghi et al., 2019; Yi, 2020) while



(a) Matrix $\mathbf{A}$ (b) Matrix $\mathbf{B}$

Fig. 3.1.: Illustration of permutation described in (Henderson and Searle, 1981). The matrix $\mathbf{A}$, with blocks where each block has constant diagonal and sub-diagonal values, can be reshaped into a block diagonal form $\mathbf{B}$ using permutation matrices $\mathbf{P}_{\text{in}}$ and $\mathbf{P}_{\text{out}}$, preserving singular values. The figure is inspired by Gnassounou et al. (2024).

preserving the singular values (Henderson and Searle, 1981); see Fig. 3.1. As a result, the largest singular value $\sigma_1(\mathbf{W}_{\text{circ}})$ can be obtained directly from the block-diagonal matrix $\mathbf{D}$.

Let $(\mathbf{D}_i)_{i=1,\dots,n^2}$ be the diagonal blocks of the matrix $\mathbf{D}$, then:

$$\sigma_1(\mathbf{W}_{\text{circ}}) = \sigma_1(\mathbf{D}) = \max_i \sigma_1(\mathbf{D}_i) \ . \tag{3.14}$$

It requires calculating the spectral norm of each $\mathbf{D}_i$, one could use SVD as in Sedghi et al. (2019) or PI as in Yi (2020), however, these algorithms do not scale up nor are efficient. In the following section, we present a more precise, fast, and scalable approach based on previously introduced Gram iteration.

3.5 Spectral norm of circular padding convolutional layers

The spectral norm of circular convolutional layers can be estimated using GI with Equation (3.14). The method applies Theorem 3.2.4 to sequences $(\mathbf{D}_i^{(t)})$, derived via a 2D Fast Fourier Transform (FFT2) of the convolutional filter K.

Proposition 3.5.1.

The spectral norm of a circular convolutional layer is upper-bounded as:

$$\sigma_1(\mathbf{W}_{\text{circ}}) \leq \max_{1\leq i\leq n^2} \|\mathbf{D}_i^{(t)}\|_{\text{F}}^{2^{1-t}} \underset{t\to+\infty}{\longrightarrow} \sigma_1(\mathbf{W}_{\text{circ}}) \,. \tag{3.15}$$

Here, $\|\mathbf{D}_i^{(t)}\|_{\text{F}}^{2^{1-t}}$ is the 2^t-Schatten norm, which is convex. The maximum over these norms defines a convex and subdifferentiable upper bound. Using Proposition 3.5.1,

Algorithm 5 : norm2_circ(K, N_{iter})

Inputs: Filter K, iterations N_{iter}
$\mathbf{D} \leftarrow \texttt{FFT2}(K)$ ▷ Apply 2D FFT to the filter
$r \leftarrow 0_{n^2}$ ▷ Initialize rescaling factors
for $1 \dots N_{\text{iter}}$
  for $i = 1 \dots n^2$ ▷ Parallel execution
    $r_i \leftarrow 2(r_i + \log\|\mathbf{D}_i\|_{\text{F}})$
    $\mathbf{D}_i \leftarrow \mathbf{D}_i / \|\mathbf{D}_i\|_{\text{F}}$ ▷ Rescale to prevent overflow
    $\mathbf{D}_i \leftarrow \mathbf{D}_i^* \mathbf{D}_i$ ▷ Gram iteration step
$\sigma_1 \leftarrow \max_i\{\|\mathbf{D}_i\|_{\text{F}}^{2^{-N_{\text{iter}}}} \exp(2^{-N_{\text{iter}}} r_i)\}$
return σ_1

Algorithm 5 computes the spectral norm of a circular convolutional layer represented by K. The method is deterministic and precise due to the properties of Gram iteration and scales efficiently by handling n^2 small matrices of size $c_{\text{out}} \times c_{\text{in}}$ in parallel. For large input sizes, using a reduced spatial dimension $n_0 \leq n$ can further reduce computation while maintaining an upper bound, as detailed in Appendix 3.7.2. For the backward computation, we use the previous Proposition 3.2.5 on each bound derived from the $\mathbf{D}_i$ for each i. The 2D FFT step is quite efficient, with complexity $O(n \log_2 n)$, making the method practical for small to moderate n. Approximations between circular and zero padding are discussed in Appendix 3.7.1. In the following section, we present an application of GI to estimate directly the spectral norm of convolutional layers with zero padding.

3.6 Spectral norm of zero padding convolutional layers

In practical implementations, zero padding is commonly used to preserve the spatial dimensions of the input and output. It is the default padding in most deep learning frameworks and convolutional neural network (CNN) architectures. For a convolutional layer with zero padding, the operation can be represented by a banded doubly block Toeplitz matrix $\mathbf{W}_{\text{toep}}$, formed by concatenating structured blocks. This matrix, of size $n^2 c_{\text{out}} \times n^2 c_{\text{in}}$, is too large for direct spectral norm computation.

Most approaches to spectral norm estimation in convolutional layers assume circular padding due to the convenient properties of the Fourier transform, which simplifies the computation. However, the zero-padding case is more challenging, as it does not share the same spectral properties.

To handle the large matrix $\mathbf{W}_{\text{toep}}$, we follow the convolutional operator approach of Wang et al. (2020) and Prach and Lampert (2022, Appendix A.1). In particular, Prach and Lampert (2022) introduce a rescaling along the output channel dimension j in order to obtain Lipschitz convolutional layers:

$$r_j = \sum_{i_1,k',l'} \left| \sum_{j=1}^{c_{\text{out}}} K_{j,i_1} \star K_{j,i_2} \right|_{k',l'}.$$

The rescaled kernel is then defined as

$$K_{\text{rescaled}} = K \star R, \qquad R \in \mathbb{R}^{1\times 1\times c_{\text{out}}\times c_{\text{out}}}, \quad R_{1,1,j,j} = r_j.$$

This rescaling can be interpreted as performing a single Gram iteration on the filter K. We propose to generalize this idea to an arbitrary number of iterations, thereby establishing a new bound on the spectral norm of $\mathbf{W}_{\text{toep}}$.

Theorem 3.6.1 (Bound on the spectral norm of zero-padded convolutional layers)**.** *For a zero-padded convolutional layer $\mathbf{W}_{\text{toep}}$ parametrized by a filter $K \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}\times k\times k}$, let us note the Gram iterate for matrix $\mathbf{W}_{\text{toep}}$ as ${\mathbf{W}_{\text{toep}}}^{(t)}$. We define the Gram iterate for filter as $K^{(t+1)}_{i_1,i_2} = \sum_{j=1}^{c_{\text{out}}} K^{(t)}_{j,i_1} \star K^{(t)}_{j,i_2}$, with convolution done with maximal non-trivial zero padding, with $K^{(1)} = K$.*
For norm $\|\cdot\|_\infty$, we have the following bound:

$$\|{\mathbf{W}_{\text{toep}}}^{(t+1)}\|_\infty^{2^{-t}} \leq \left(\max_{i_2} \sum_{i_1,k',l'} \left| \sum_{j=1}^{c_{\text{out}}} K^{(t)}_{j,i_1} \star K^{(t)}_{j,i_2} \right|_{k',l'} \right)^{2^{-t}}.$$

For norm $\|\cdot\|_{\text{F}}$, we have:

$$\|{\mathbf{W}_{\text{toep}}}^{(t+1)}\|_{\text{F}}^{2^{-t}} \leq \left(k^2 \sum_{i_1,i_2,k',l'} \left| \sum_{j=1}^{c_{\text{out}}} K^{(t)}_{j,i_1} \star K^{(t)}_{j,i_2} \right|_{k',l'}^2 \right)^{2^{-(t+1)}},$$

and $\sigma_1(\mathbf{W}_{\text{toep}}) \leq \|{\mathbf{W}_{\text{toep}}}^{(t)}\|^{2^{-t}} \underset{t\to\infty}{\longrightarrow} \sigma_1(\mathbf{W}_{\text{toep}})$ with super geometric convergence.

The maximum non-trivial padding size is the maximum padding size, $k-1$, where the convolution kernel still interacts with the original input regions, ensuring it does not interact exclusively with zeros from the padding. Note that this bound also holds

for the case of circular padding, see proof in Appendix. Pairing with Equation (A.2), we have Algo. 6 for the choice of matrix norm $\| \, . \, \|_\infty$.

Algorithm 6 : norm2_toep(K, N_{iter})

Inputs filter: K, number of iterations: N_{iter}
$r \leftarrow 0$
for $1 \dots N_{\text{iter}}$
  $r \leftarrow 2(r + \log\|K\|_F)$
  $K \leftarrow K/\|K\|_F$
  $K \leftarrow \text{conv2d}(K, K, \text{padding} = \text{kernel\_size}(K) - 1)$
$\sigma_1 \leftarrow \max_i \left\{ \left(\sum_{j,k_1,k_2} |K_{j,i,k_1,k_2}| \right)^{2^{-N_{\text{iter}}}} \exp\left(2^{-N_{\text{iter}}} r\right) \right\}$
return σ_1

The bound may not be tight, as the use of absolute values in the summations discards potential cancellation effects between filter coefficients. This yields a conservative estimate of the spectral norm, although the Gram iteration ensures fast convergence of the bound.

The method is referred to as `norm2_toep` because it provides an upper bound on the spectral norm of convolutional layers with zero-padding, which are classically represented as Toeplitz matrices. However, it also applies to circular convolutions, where the associated matrices are circulant.

Our proposed method leverages the same principles as power iteration (Ryu et al., 2019) but with significant advantages for convolutional layers. Unlike power iteration, it is deterministic and provides a quadratically convergent certified upper bound on the spectral norm at any iteration, whereas power iteration does not guarantee this property. Specifically adapted for convolutional layers, it utilizes efficient `conv2d` operations with GPU acceleration, making it highly practical for large-scale applications.

In the following, we study the relation between the spectral norm of convolutional layers with circular and zero padding.

3.7 Bound approximations between input sizes and padding types

This section presents theoretical and practical approximations of spectral-norm bounds for convolutional layers that remain accurate while reducing cost. Exact computation scales poorly with the spatial size n: with circular padding it requires

evaluating n^2 spectral norms of the $c_{\text{in}} \times c_{\text{out}}$ matrices $\mathbf{D}_i$, and with zero padding and large kernels the Gram-iteration approach (Alg. 6) is likewise expensive.

To address these challenges, we propose an approximation that extends the computationally efficient spectral norm bounds for circular padding to the zero-padding case. While circular padding, modeled by circulant matrices, facilitates spectral norm computations due to its connection to Fourier transforms, zero padding, modeled by Toeplitz matrices, is more commonly used in practice. Section 3.7.1 and Theorem 3.7.1 establish how the spectral norm bound computed under circular padding can approximate the bound for zero padding, leveraging the structure of circulant matrices to simplify computation. In particular, for zero padding, spectral norm computations scale with the kernel size but remain independent of the input size, making this approximation particularly relevant for the large kernel size. Another consideration arises when dealing with large input sizes. Circular padding bounds are computationally efficient with respect to kernel size but become expensive as the input size n increases. Section 3.7.2 and Theorem 3.7.3 examine how spectral norm bounds computed for a smaller spatial dimension $n_0 \leq n$ can serve as a reliable approximation for larger inputs, significantly reducing computational costs.

This section presents theoretical results and practical methods for approximating spectral norm bounds in these scenarios, ensuring efficient and scalable computations for convolutional layers.

3.7.1 From circular to zero padding

In CNN architectures, zero-padded convolutions are more common than circular ones. We want to use the circular convolution theory to obtain a bound on the spectral norm of zero padding convolution.

Considering the block diagonal matrix $\mathbf{D}_i$ built from the filter K, we re-index $(\mathbf{D}_i)_{i=1,\dots,n^2}$ as $(\mathbf{D}_{u,v})_{1\leq u,v\leq n}$, where $\mathbf{W}_{\text{circ}}$ denotes the associated matrix representing the circular convolutional layer. Using the discrete Fourier transform (DFT) matrix $\mathbf{U} \in \mathbb{C}^{n\times n}$, the expression for $\mathbf{D}_{u,v}$ is given by:

$$\mathbf{D}_{u,v} = \sum_{k_1=0}^{n-1}\sum_{k_2=0}^{n-1} \mathbf{U}_{k_1,u}\, K_{:,:,k_1,k_2}\, \mathbf{U}_{k_2,v} \qquad = \sum_{k_1=0}^{k-1}\sum_{k_2=0}^{k-1} e^{-\frac{2\pi \mathbf{i} k_1 u}{n}}\, e^{-\frac{2\pi \mathbf{i} k_2 v}{n}}\, K_{:,:,k_1,k_2}\ ,$$

where K is zero-padded. Similarly, we define $\mathbf{W}_{\text{toep}}$ as the matrix constructed from the filter K, representing the zero-padded convolutional layer. We recall the bound on spectral norm produced with Gram iteration from Theorem 3.2.4:

$$\sigma_1(\mathbf{W}_{\text{circ}}) = \max_{1\leq u,v\leq n} \sigma_1(\mathbf{D}_{u,v}) \leq \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^{2^{1-t}} .$$

The following theorem establishes a bound approximation between circular and zero padding convolutional layers.

Theorem 3.7.1 (Bound approximation for zero padding)**.**
For given t and kernel size k, let

$$n \geq 2^t \left\lfloor \frac{k}{2} \right\rfloor + 1, \quad \alpha = \frac{2^t \lfloor k/2 \rfloor}{n}.$$

Then the spectral norm of the zero-padded convolutional layer can be bounded by:

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \left(\frac{1}{1-\alpha}\right)^{2^{-t}} \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^{2^{1-t}} .$$

See proof in Appendix A.0.6. This result can be used when computing the bound on the spectral norm of convolution layers with circular padding is more convenient than zero padding, namely because circulant matrix theory allows the use of Fourier transform to ease computation. It can be crucial in the case of large kernel size k.

Orthogonal convolutional operators (Li et al., 2019; Singla and Feizi, 2021c; Trockman and Kolter, 2021; Yu et al., 2022) allow for spectral norm control, particularly in the case of circular padding.

Proposition 3.7.2 (Contractant property of orthogonal convolutional layers with zero padding)**.**
For orthogonal convolutional layers with circular padding and filter K, we have:

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \sigma_1(\mathbf{W}_{\text{circ}}) = 1 ,$$

where $\sigma_1(\cdot)$ denotes the spectral norm of the corresponding matrix.

See proof in Appendix A.0.7. This property extends to zero-padding convolutional layers by leveraging the relationship between their spectral norms. Specifically, contractive zero-padding convolutional layers can be derived from orthogonal circular padding layers, ensuring $\sigma_1(\cdot) \leq 1$. By applying orthogonalization techniques the stability and 1-Lipschitz constraint of circular padding convolutional layers can be

transferred to the zero-padding setting, facilitating the construction of orthogonal neural networks in the standard zero-padding setting.

3.7.2 Lower input size approximation

We distinguish between the multichannel filter padded to match the input size n, denoted as $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times n \times n}$, and the filter padded to a reduced input size n_0, where $k \ll n_0 \leq n$, denoted as $K^{\downarrow} \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times n_0 \times n_0}$. In the same manner, we define $(\mathbf{D}^{\downarrow}_{u,v})_{1 \leq u,v \leq n}$ as the block diagonal matrices constructed from $K^{\downarrow}$, with the corresponding circular convolution matrix denoted as $\mathbf{W}_{\text{circ}}{}^{\downarrow}$. A similar expression holds for $\mathbf{D}^{\downarrow}_{u,v}$:

$$\mathbf{D}^{\downarrow}_{u,v} = \sum_{k_1=0}^{k-1} \sum_{k_2=0}^{k-1} e^{-\frac{2\pi \mathbf{i} k_1 u}{n_0}} \, e^{-\frac{2\pi \mathbf{i} k_2 v}{n_0}} \, K_{:,:,k_1,k_2} \ .$$

Similarly, the matrix $\mathbf{W}^{\downarrow}_{\text{toep}}$ is constructed from $K^{\downarrow}$, representing the zero-padded convolutional layer at spatial resolution n_0.

To consider a very large input spatial size n, we can use our bound by approximating the spatial size for $n_0 \leq n$. It means we pad the kernel K to match the spatial dimension $n_0 \times n_0$, instead of $n \times n$. To compensate for the error committed by the sub-sampling approximation, we multiply the bound by a factor α. The work of (Pfister and Bresler, 2019) analyzes the quality of approximation depending on n_0 and gives an expression for factor α to compensate and ensure to remain an upper-bound, as studied in (Araujo et al., 2021).

We can estimate the spectral norm bound for a convolution defined by kernel K and input size n with the spectral norm bound for a lower input size $n_0 \leq n$.

Theorem 3.7.3 (Bound approximation for lower input size)**.**
For given t and kernel size k, let

$$n_0 \geq 2^t \left\lfloor \tfrac{k}{2} \right\rfloor + 1, \quad \alpha = \frac{2^t \lfloor k/2 \rfloor}{n_0}.$$

the spectral norm of the circular convolutional layer can be bounded by:

$$\sigma_1(\mathbf{W}_{\text{circ}}) \leq \left(\frac{1}{1-\alpha} \right)^{2^{-t}} \max_{1 \leq u,v \leq n_0} \|\mathbf{D}^{\downarrow (t)}_{u,v}\|_{\text{F}}^{2^{1-t}} \ .$$

Same bound holds for zero padding convolutional layers:

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \left(\frac{1}{1-\alpha} \right)^{2^{-t}} \max_{1 \leq u,v \leq n_0} \|\mathbf{D}^{\downarrow (t)}_{u,v}\|_{\text{F}}^{2^{1-t}} \ .$$

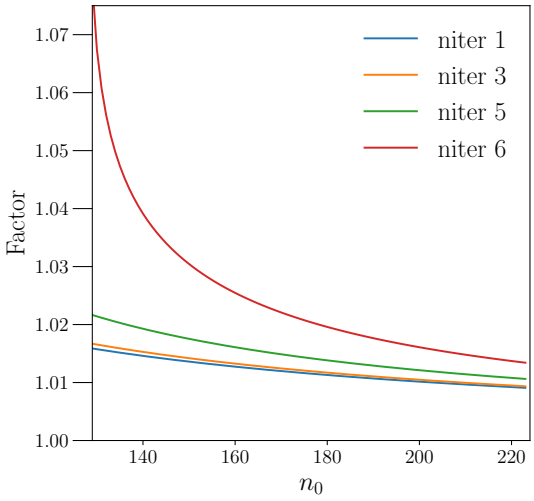


Fig. 3.2.: Evolution of bound factor $(1/(1-\alpha))^{2^{-t}}$, described in Theorem 3.7.3, for input size $n = 224$, kernel size $k = 3$ and number of Gram iterations $t \in \{1, 3, 5, 6\}$.

We illustrate the evolution of the factor term $\left(\frac{1}{1-\alpha}\right)^{2^{-t}}$ in Figure 3.2 for different n_0 and for different number of Gram iteration t. We see that the more the number of Gram iterations t increases the larger the factor, however, the estimation of the spectral norm is also much more accurate (as the bound is tighter). For a very large image size n, this theorem allows the computation of a tight bound on the spectral norm of a convolutional layer at a low computational cost w.r.t input spatial size.

3.8 The case of strided convolutions

Convolutional layers with stride $s > 1$ induce a downsampling effect that reduces the output resolution and modifies the associated linear operator. Let $\mathbf{W}_{K,s}$ denote the operator corresponding to a convolution with kernel K and stride s. When $s = k$ (i.e., non-overlapping convolutions), the spectral norm of the convolution is equal to that of the kernel:

$$\|\mathbf{W}_{K,s}\|_2 = \|K\|_2.$$

In general, the spectral norm of a strided convolution can be upper bounded as

$$\|\mathbf{W}_{K,s}\|_2 \leq \|\text{DownSampling}_s\|_2 \cdot \|\mathbf{W}_{K,1}\|_2,$$

where DownSampling_s is the operator that selects one entry every s positions. Since this operator is contractive, we have $\|\text{DownSampling}_s\|_2 \leq 1$, implying that stride reduces the spectral norm.

In practice, estimating the spectral norm of $\mathbf{W}_{K,s}$ can be done directly by applying power iteration to the convolution operator with stride s, i.e., using `conv2d` with

`stride=s`. For a more refined analysis, one can adapt the Gram iteration procedure to take into account the downsampling effect, as proposed by Senderovich et al. (2022).

3.9 Experiments

For research reproducibility, the code of experiments is available at `https://github.com/blaisedelattre/lip4conv`. Experiments were done on one NVIDIA RTX A6000 GPU.

3.9.1 Spectral norm computation for dense layers

This experiment aims to compare the convergence behavior of Power iteration (PI) and Gram iteration (GI) in computing the spectral norm of dense layers, specifically focusing on matrices.

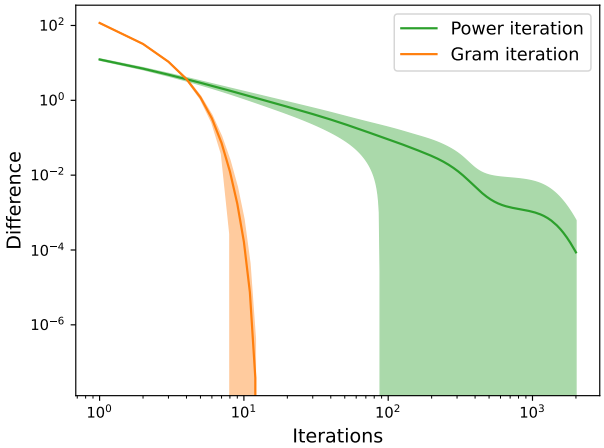


Fig. 3.3.: Convergence plot in log-log scale for spectral norm computation, comparing Power iteration and Gram iteration, one standard deviation shell is represented in a light color. Difference at each iteration is defined as $|\sigma_{1\text{method}} - \sigma_{1\text{ref}}|$. Gram iteration converges to numerical 0 in less than 12 iterations while Power iteration has not yet converged with 2,000 iterations and has a larger dispersion through runs.

We compare PI and GI methods for spectral norms computation, and a set of 100 random Gaussian matrices 2000×1000 is generated. The reference value for the spectral norm of a matrix $\sigma_{1\text{ref}}$ is obtained from PyTorch using SVD, and Algorithm 4 is used for GI. Estimation error is defined as $|\sigma_{1\text{method}} - \sigma_{1\text{ref}}|$. We observe in Figure 3.3 that PI needs more than 2,000 iterations to fully converge to numerical 0 and at minimum 90 iterations. Runs have a much larger standard deviation in comparison to GI. For that experiment, full convergence required up to 5,000 iterations for some realizations. This highlights how PI convergence can be very slow

from one run to another for different generated matrices. Furthermore, in Figure 3.5 in Appendix we show convergence of methods when a generated matrix is fixed: variance PI observed for PI is only due to its non-deterministic behavior, unlike GI which is deterministic. This shows how the variance of PI is sensitive due to two causes: random generation of input matrices and the method in itself with random vector initialization at the start of the algorithm. On the contrary, GI converges under 15 iterations in every case.

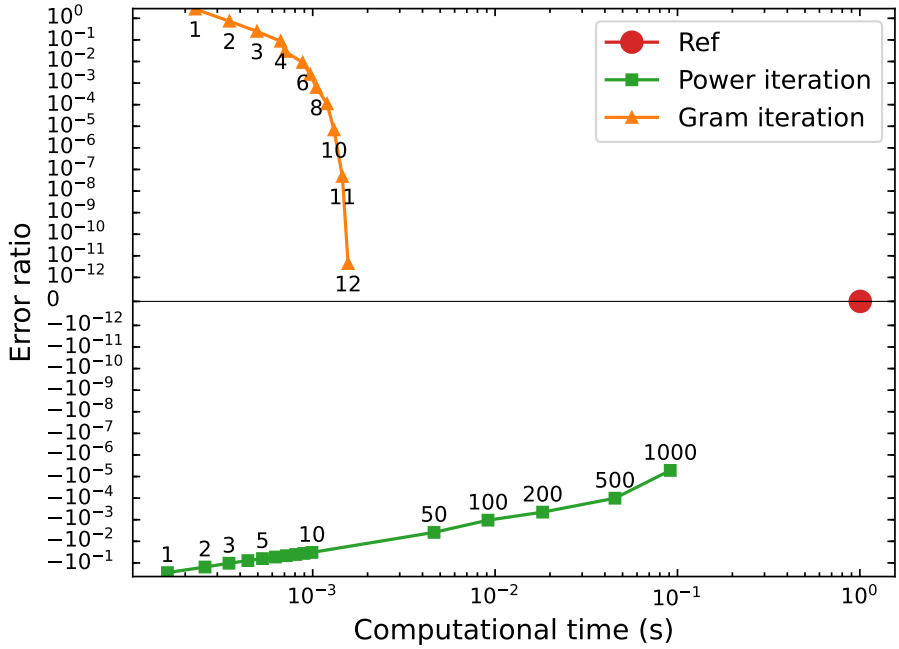


Fig. 3.4.: Error ratios over computational times of methods for spectral norm computation. Error ratio is defined as $\sigma_{1\text{method}}/\sigma_{1\text{ref}} - 1$, PyTorch SVD is taken as a reference. Points are annotated with the number of iterations. GI converges in $\pm 10^{-3}$s to numerical 0, while PI convergence is slower.

To capture the sign of the error, *i.e.* if the current estimator is an upper bound or not, we define the error ratio as $\sigma_{1\text{method}}/\sigma_{1\text{ref}} - 1$, depicted in Figure 3.4 also reported in Table A.0.1. We observe that the error ratio for PI is negative at each iteration, meaning that PI estimation approaches the reference while being inferior to it, thus being a lower bound. GI empirically enjoys a very fast convergence, with 10 iterations: doing one more iteration divide the error ratio by approximately $1e2$ and two more iterations by $1e4$. In contrast, PI has a much slower convergence: passing from 10 to 100 iterations only divides the ratio by a factor of 10. This faster convergence rate from GI makes the method much faster in terms of computational time than PI and SVD for the same precision. Those results motivate the use of GI for dense layers as well.

Experiment: PI vs. GI with different readouts. We compare Gram iteration (GI) with different choices of final readout norm against the classical Power Iteration (PI); see

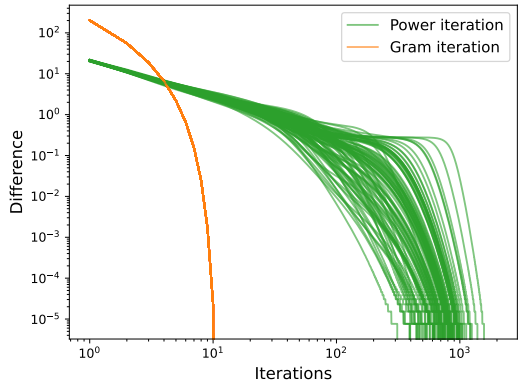


Fig. 3.5.: Convergence plot in log-log scale for spectral norm computation, comparing power iteration and Gram iteration, the same matrix is used here, each line corresponds to one run of Power iteration and Gram iteration. The difference is defined as $|\sigma_{1\text{method}} - \sigma_{1\text{ref}}|$ where reference is taken from PyTorch. We see that Power iteration has inherent randomness by design whereas Gram iteration is fully deterministic.

Fig. 3.6. The goal is to highlight how the choice of norm in the final evaluation of $s_t = \|\mathbf{W}^{(t)}\|^{2^{1-t}}$ impacts the asymptotic rate.

The test matrix is constructed as

$$\mathbf{W} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)\mathbf{V}^\top,$$

where $\mathbf{U}, \mathbf{V}$ are independent Haar–distributed orthogonal matrices (obtained from the QR factorization of Gaussian matrices). We fix $\sigma_1 = 1$, $\sigma_2 = \rho \in \{0.6, 0.8\}$ to control the spectral gap, and let the remaining singular values decrease geometrically. Unless otherwise stated we set $p = q = 60$ and keep the random seed fixed to ensure reproducibility.

GI follows Algorithm 4: at each step the iterate is rescaled by the Frobenius norm to avoid overflow, while the final readout is taken with $\|\cdot\|_\star \in \{\|\cdot\|_F, \|\cdot\|_1, \|\cdot\|_\infty\}$. PI is run on $\mathbf{A} = \mathbf{W}^*\mathbf{W}$, with iterates $\mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k/\|\mathbf{A}\mathbf{v}_k\|_2$ and Rayleigh quotient estimate $s_k^{(\mathrm{PI})} = \sqrt{\mathbf{v}_k^\top \mathbf{A}\mathbf{v}_k}$. We measure the relative error $e_t = |s_t - \sigma_1(\mathbf{W})|/\sigma_1(\mathbf{W})$ across iterations, as well as the local order $p_t = \log(e_{t+1})/\log(e_t)$. This allows us to diagnose whether the observed convergence is linear, superlinear, or close to quadratic.

The results clearly separate the methods. With Frobenius readout, GI achieves supergeometric decay $e_t = \Theta(q^{2^t}/2^t)$ and local order $p_t \uparrow 2$, giving an empirical convergence nearly indistinguishable from quadratic (although not R–quadratic in theory). In contrast, GI with $\|\cdot\|_1$ or $\|\cdot\|_\infty$ readout converges much more slowly,

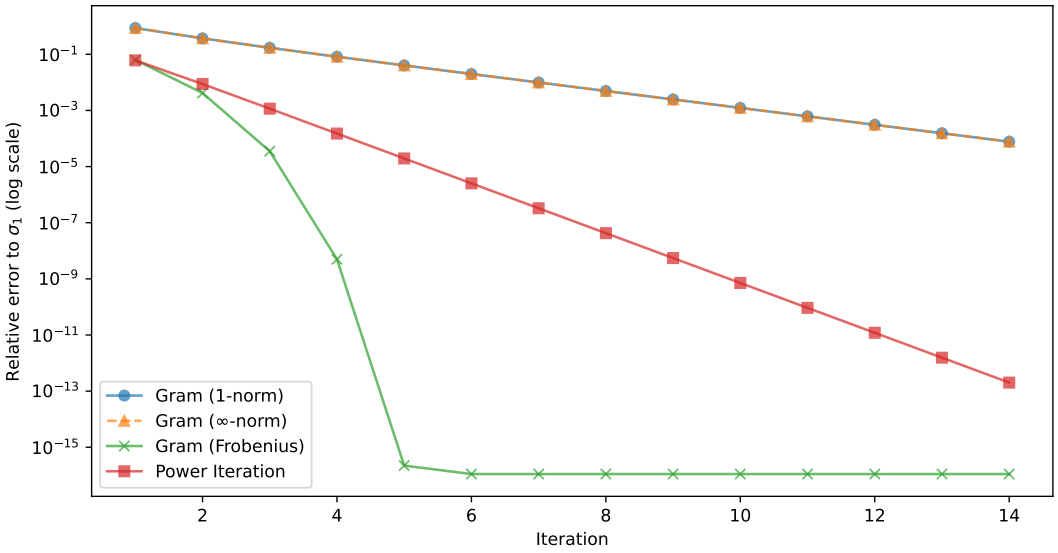


Fig. 3.6.: Comparison of Gram iteration with different final readouts (Frobenius, $\|\cdot\|_1$, $\|\cdot\|_\infty$) and Power Iteration (PI). Only the Frobenius readout exhibits *supergeometric* convergence with local order tending to 2 (near–quadratic appearance), consistent with $e_t = \Theta(q^{2^t}/2^t)$. Other readouts converge more slowly (effectively R–linear), while PI converges linearly at rate σ_2/σ_1.

consistently with an R–linear bound. PI recovers its classical behavior, converging linearly with rate $q = \sigma_2/\sigma_1$.

3.9.2 Estimation for circular-padded convolutions

Estimation on simulated convolutional layers This experiment evaluates the performance of various Lipschitz-bound methods in terms of precision and computational time. We consider convolutional layers with constant padding and vary three parameters: number of input and output channels (Figure 3.8), input size n (Figure 3.9), and kernel size k (Figure 3.7). For each experiment, random kernels are sampled from Gaussian and uniform distributions, repeated 50 times, with results averaged for both errors ($\sigma_{1\text{method}} - \sigma_{1\text{Ryu2019}}$) and computational time.

The methods compared include Ryu et al. (2019) (100 iterations for precise estimates), Singla and Feizi (2021a) (50 iterations as per their code), Araujo et al. (2021) (50 samples), and our Algorithm 5 (5 iterations). Our method achieves the same spectral bound as Sedghi et al. (2019) under the circular convolution hypothesis, while being the fastest among all methods.

We observe that Ryu et al. (2019)'s computational cost scales significantly with n and k, whereas other methods based on the circular convolution hypothesis have near-constant computational times. As kernel size k increases, the difference between circular and zero padding becomes more significant. These results high-

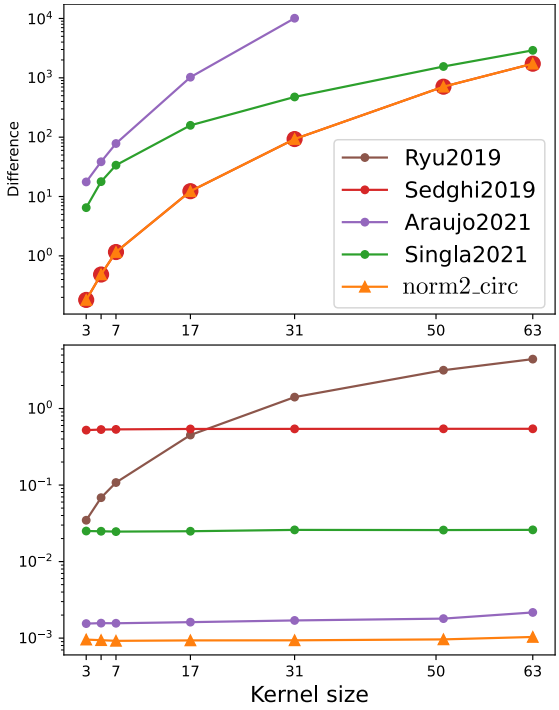


Fig. 3.7.: Effect of varying kernel size k with $c_{\text{in}} = 16$, $c_{\text{out}} = 16$, and $n = 252$. The difference is expressed as $\sigma_{1\,\text{method}} - \sigma_{1\,\text{Ryu2019}}$. As k increases, the Lipschitz difference between circular and zero padding becomes more pronounced. Our method outperforms others in speed and precision, matching Sedghi et al. (2019). On the y-axis of the first plot, the difference is expressed as $\sigma_{1\,\text{method}} - \sigma_{1\,\text{Ryu2019}}$. On the second plot, the computational time is expressed in seconds.

light the scalability and precision of our approach for varying convolutional layer parameters.

Estimation on real convolutional layers In this experiment, we use the method of Ryu et al. (2019) with 100 iterations (to ensure convergence) as the reference for spectral norm computation. Since most convolutional layers in CNNs use constant zero padding, we also compare with Sedghi et al. (2019), which provides the exact spectral norm under the circular padding assumption.

Inspired by Singla and Feizi (2021a), we estimate the spectral norm for each convolutional layer of a ResNet18 (He et al., 2016) pre-trained on the ImageNet-1k dataset. Figure 3.10 shows the difference between the reference and the spectral norm bounds for various methods across all convolutional layers.

Our method produces bounds very close to the exact values given by Sedghi et al. (2019), while consistently remaining above the reference value of Ryu et al. (2019), ensuring a certified upper bound. In contrast, we observe that for filter index 3, the method of Singla and Feizi (2021a) reports a value of 2.03, which is below the reference value of 2.05. This highlights the limitations of using power iteration (PI)

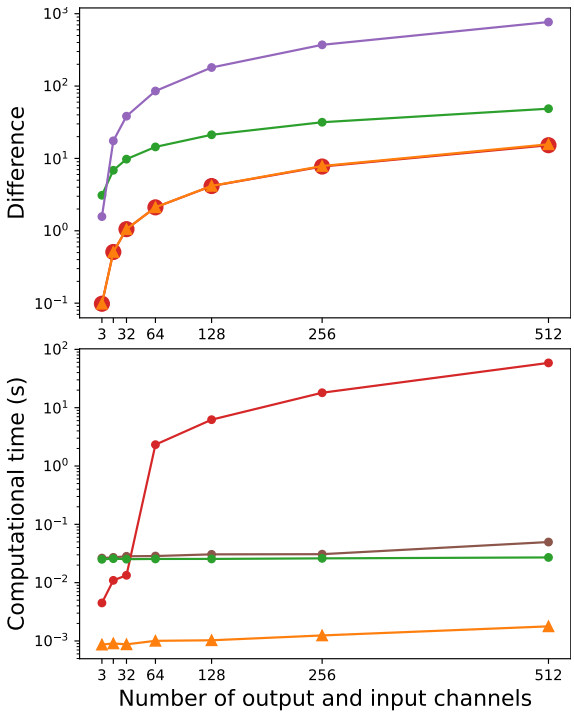


Fig. 3.8.: Effect of varying the number of channels c_{in} and c_{out} with fixed input size $n = 32$ and kernel size $k = 3$. The difference is expressed as $\sigma_{1\text{method}} - \sigma_{1\text{Ryu2019}}$. Our method achieves the best balance between speed and precision, matching the values from Sedghi et al. (2019), while Ryu et al. (2019) scales strongly with input and kernel size. On the y-axis of the first plot, the difference is expressed as $\sigma_{1\text{method}} - \sigma_{1\text{Ryu2019}}$. On the second plot, the computational time is expressed in seconds.

in the pipeline to obtain a reliable upper bound. Detailed numerical results are provided in Table A.0.3.

3.9.3 Estimation for zero-padded convolutions

This experiment evaluates the accuracy and computational cost of spectral norm estimation methods for zeros-padded convolutional layers. While the case of circular padding has been addressed in the previous subsection, it does not provide theoretical guarantees for zeros-padded convolutional layers.

We exclude the method of Sedghi et al. (2019) from this comparison because our Gram iteration-based circulant method (norm2_circ) achieves the same reference values for circular padding in significantly less computational time. For ground truth, we generate random filters K sampled from a Gaussian distribution and construct the corresponding doubly banded Toeplitz matrix, as described in Section 3.3. The spectral norm of this matrix is then computed using NumPy's matrix_norm function.

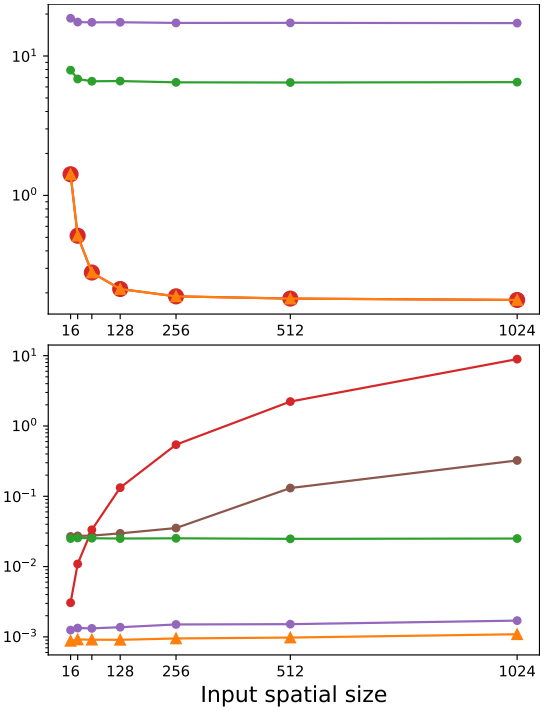


Fig. 3.9.: Effect of varying input size n, with $c_{\text{in}} = 16$, $c_{\text{out}} = 16$, and $k = 3$. The difference is expressed as $\sigma_{1\,\text{method}} - \sigma_{1\,\text{Ryu2019}}$. Our method is efficient and precise, comparable to Sedghi et al. (2019), while Ryu et al. (2019) exhibits high computational costs for larger n. On the y-axis of the first plot, the difference is expressed as $\sigma_{1\,\text{method}} - \sigma_{1\,\text{Ryu2019}}$. On the second plot, the computational time is expressed in seconds.

For Gram iteration-based methods, we use six iterations, while the power iteration method of Ryu et al. (2019) is run with 100 iterations to ensure convergence. We limit our tests to $1 \leq c_{\text{in}}, c_{\text{out}} \leq 32$ to avoid memory constraints associated with the large size of the Toeplitz matrix. The circulant approach (norm2_circ) incorporates a corrective factor α, ensuring an upper bound on the spectral norm, as demonstrated in Theorem 3.7.1, to make it applicable for zeros-padded convolutional layers.

Figure 3.11 shows that the power iteration method fails to guarantee an upper bound on the spectral norm. In particular, the method of Ryu et al. (2019) consistently underestimates the true spectral norm. By contrast, our norm_toep method delivers the best accuracy with a low computational cost, demonstrating its effectiveness for zeros-padded convolutional layers.

3.9.4 Memory impact of gradient computation

Memory cost of `norm2_circ`. For the case of norm2_circ, naively applying automatic differentiation to estimate the gradient of the spectral norm bound via Gram iteration leads to a high memory footprint, particularly when FFT-based convolution is used.

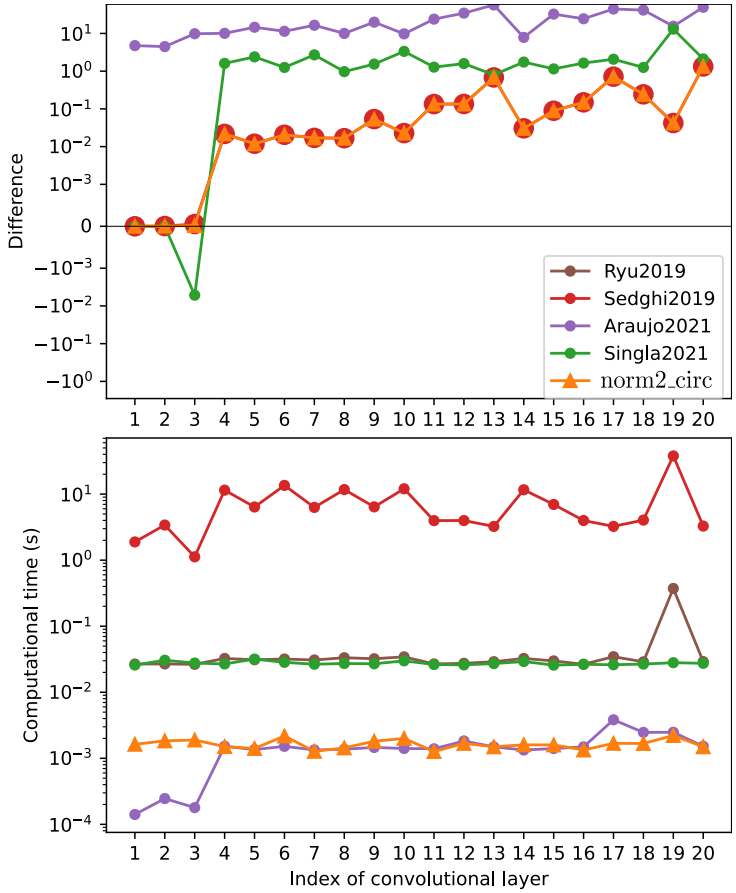


Fig. 3.10.: Comparison of spectral norm bounds for all convolutional layers of a pre-trained ResNet18. The reference value is obtained using the method of Ryu et al. (2019). Each convolutional layer in ResNet18 has varying numbers of input and output channels, kernel sizes, and input spatial dimensions. This experiment evaluates the bounds on real kernel filters, demonstrating that our method achieves the best precision while maintaining comparable computational times to Araujo et al. (2021).

This is because autodiff stores all intermediate activation maps for the backward pass, which becomes prohibitive when regularizing all convolutional layers of deep networks. Instead, we derive the gradient explicitly (using Proposition 3.2.5), avoiding backpropagation through the full iterative Gram procedure.

Table 3.2 compares the peak GPU memory usage when using autodiff versus our explicit formulation. On high-resolution inputs (224×224), the explicit method reduces memory usage by more than $2\times$ on ResNet18/34 and remains feasible on ResNet50, whereas autodiff exceeds available memory.

Memory cost of `norm2_toep`. The `norm2_toep` method estimates the spectral norm of a convolutional filter via recursive Gram iterations, where each iteration involves a self-convolution of the kernel. To preserve spatial alignment, zero-padding of size $k-1$ is applied at every iteration, where k is the initial kernel size. This results in a

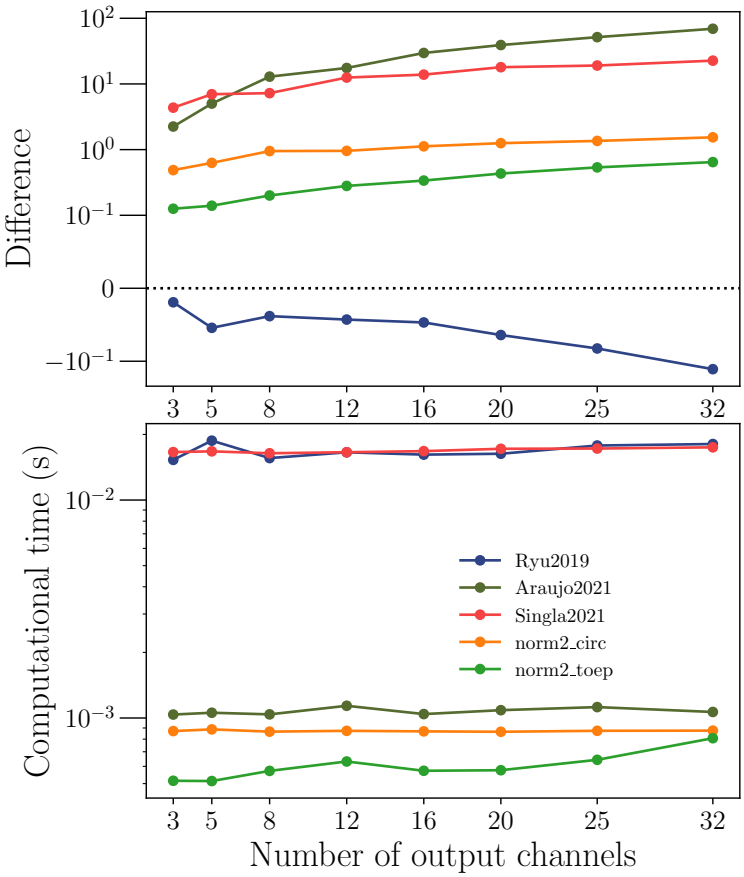


Fig. 3.11.: Estimation error and computational time for spectral norm computation of zeros-padded convolutional layers with varying numbers of input and output channels $(c_{\text{in}}, c_{\text{out}})$. Kernel size is $k = 3$ and input size is $n = 32$. The reference value for the spectral norm is taken as ground truth by computing the spectral norm of the corresponding Toeplitz matrix with NumPy's matrix_norm function. The results compare different state-of-the-art methods.

progressive expansion of the kernel's spatial support. After t Gram iterations, the effective kernel size becomes:

$$s_t = (k-1)\,t + 1\,.$$

Hence, the memory required to store the filter grows quadratically with the number of iterations:

$$\#\text{elements}_t = s_t^2 = \big((k-1)\,t + 1\big)^2\,.$$

This quadratic growth imposes practical constraints, especially for large t or wide filters. In practice, moderate values such as $t = 6$ remain tractable. Nevertheless, the overall memory usage of `norm2_toep` is typically lower if convolutional filters are usually small (e.g., $k = 3$); second, fewer iterations are needed to obtain a reliable upper bound.

Tab. 3.2.: Peak GPU memory (in MB) during training with Gram regularization on all convolutional layers ($t = 6$ iterations).

Model	Autodiff	Explicit Gradient
ResNet18	9770	**4322**
ResNet34	17152	**7088**
ResNet50	> 42000	**27535**

3.10 Conclusion

In this chapter, we introduced novel methods based upon Gram iteration for spectral norm estimation that are deterministic, near quadratically convergent (super geometric in theory), differentiable, and consistently provide certified upper bounds. These methods are designed for three specific cases:

- Dense layers represented by dense matrices.
- Circularly padded convolutional layers modeled as concatenations of doubly block circulant matrices.
- Zeros-padded convolutional layers modeled as concatenations of doubly block Toeplitz matrices.

These methods leverage the structural properties of the matrices involved, ensuring computational efficiency while maintaining theoretical guarantees. They can be used to have a finer estimate of the spectral norm of linear layers in neural networks. The code is available at this link `lip4conv`.
The proposed approaches are relevant in domains where grounded upper bounds are essential, primarily in robustness certification, stability analysis, and adversarial defense. Additionally, they play a crucial role in normalizing flow architectures, where layer inversion is required (Behrmann et al., 2019) and ensuring a tight bound on the Lipschitz constant is critical for preserving invertibility and numerical stability.
As differentiable techniques, they can be seamlessly integrated into neural network training pipelines, enabling their use for regularization. This is particularly useful in enforcing Lipschitz constraints or improving the robustness and stability of models. Furthermore, we analyze the relationship between circular and zero padding, providing a theoretical approximation that allows spectral norm estimation under zero padding using results from circular padding. These findings enhance the understanding of convolutional layers' spectral properties.
Recently, the work of Boissin et al., 2025 and the package `orthogonium` have ex-

panded the design of orthogonal convolutional layers, making them more competitive in deep learning architectures. Our theoretical results can be leveraged to provide stronger guarantees on the spectral norm of these layers, ensuring more reliable control over their Lipschitz properties. Furthermore, our methods facilitate the extension of spectral norm control from circular padding to the more commonly used zero-padding layers, broadening their practical applicability.

In the next chapter, we will apply these spectral norm estimation techniques to enforce Lipschitz constraints in neural networks, demonstrating their impact on robustness, generalization, and stability in deep learning.

4 Lipschitz networks

Contents

Deep learning models often exhibit sensitivity to input perturbations (Szegedy et al., 2013), making stability a crucial consideration for both theoretical analysis and practical deployment. Lipschitz continuity provides a fundamental framework for quantifying and controlling this stability, playing a key role in generalization (Bartlett et al., 2017) and adversarial robustness (Tsuzuku et al., 2018). However, designing Lipschitz networks remains a significant challenge, as their Lipschitz constant is difficult to compute and constrain, particularly in deep architectures. A major gap persists between the performance of state-of-the-art architectures and the theoretical guarantees offered by Lipschitz networks. While Lipschitz continuity has been leveraged to improve robustness and certified guarantees, existing approaches often impose strong constraints on network expressivity, leading to a trade-off between stability and performance. This calls for improved methods to estimate, regularize, and construct Lipschitz networks that balance theoretical guarantees with practical effectiveness.

This chapter presents our contributions to the study of Lipschitz continuity in neural networks and introduces theoretical and practical tools that will be leveraged in subsequent Chapter 5 and Chapter 7. We begin by exploring methods for estimating the global Lipschitz constant of convolutional neural networks using Product Upper Bound (PUB), which provides an efficient approach to computing this Lipschitz bound. We then introduce novel regularization techniques designed to directly control the Lipschitz constant of convolutional layers during training. To complement these methods, we propose approaches for constructing neural network layers that are inherently Lipschitz continuous. Specifically, we develop a new rescaling method

Tab. 4.1.: Comparison of networks bound Lipschitz ratio with standard deviation for several CNNs, reference for computing the ratio is the bound given by Ryu et al. (2019) method. Overall Lipschitz constant bound $\mathrm{PUB}_{\text{method}}$ is estimated for each method. Ratio of network Lipschitz bound is defined as $\mathrm{PUB}_{\text{method}}(f)/\mathrm{PUB}_{\text{Ryu2019}}(f)$. Results are averaged over 100 runs. Ratio standard deviations of ours, Sedghi et al. (2019) are induced by Ryu et al. (2019) method. We give the same ratio as Sedghi et al. (2019) in a much lower time.

Model	**Ratio of Network Lipschitz Bound (Total Running Time (s))**			
	Ours	Singla and Feizi	Sedghi et al.	Ryu et al.
VGG16	1.14 ± 0.02 (0.100)	23.90 ± 6.80 (0.47)	1.14 ± 0.020 (525)	(0.64)
VGG19	1.16 ± 0.005 (0.110)	30.63 ± 0.30 (0.53)	1.16 ± 0.005 (639)	(0.71)
ResNet18	1.47 ± 0.007 (0.039)	87.93 ± 0.88 (0.50)	1.47 ± 0.007 (185)	(0.71)
ResNet34	1.82 ± 0.35 (0.060)	4982 ± 4894 (0.88)	2.15 ± 0.35 (237)	(1.16)
ResNet50	1.68 ± 0.35 (0.100)	3338 ± 4622 (1.05)	1.67 ± 0.35 (377)	(1.49)
ResNet101	1.74 ± 0.32 (0.173)	4026 ± 4178 (1.50)	1.74 ± 0.32 (551)	(2.20)
ResNet152	1.92 ± 0.46 (0.260)	$8.39\text{e+}4 \pm 1.6\text{e+}5$ (2.05)	1.92 ± 0.460 (725)	(3.01)

spectral rescaling tailored for Lipschitz networks. Finally, we present new theoretical results on the regularity of the Weierstrass transform, highlighting its applications in designing Lipschitz networks.

4.1 Lipschitz constant estimation with PUB

Building on the work presented in the chapter on spectral norm computation 3 for neural network layers, we aim to refine the Product Upper Bound (PUB) approximation for neural networks. By leveraging precise spectral norm calculations, using Gram iteration introduced in Chapter 3 (Equation 3.1), we demonstrate improvements over state-of-the-art methods for estimating the Lipschitz constant through PUB, particularly for convolutional networks. For our experiments, we use the method proposed by Ryu et al. (2019) with 100 iterations to ensure convergence as a reference value. This serves as a benchmark for comparison, given its established accuracy. It is worth noting that most convolutional layers in CNNs employ constant zero padding, whereas Sedghi et al. (2019) provides an exact spectral norm computation specifically for convolutional layers with circular padding. To compute the PUB for a CNN or ResNet architecture we follow the rule stated in the Appendix B.1. The overall Lipschitz constant of the network is assessed here using PUB. We consider several CNNs pre-trained on the ImageNet-1k dataset: VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016) of different sizes. Using rules from Appendix 4.1 to compute the Lipschitz constant of each individual layer in CNNs, we compute bound $\mathrm{PUB}(f)$ on the overall Lipschitz constant of the network. Then we

compare this produced bound for each method to the one obtained by (Ryu et al., 2019) by considering the ratio: $\mathrm{PUB}_{\text{method}}(f)/\mathrm{PUB}_{\text{Ryu2019}}(f)$.

We take 100 iterations for Ryu et al. (2019) to have a precise reference, 50 iterations for (Singla and Feizi, 2021a), 20 samples for (Araujo et al., 2021) and 7 iterations for our method, as the task requires increased precision. Results are reported in Table 4.1, we see that our method gives results similar to (Sedghi et al., 2019) and overall network Lipschitz bound is tighter than Singla and Feizi (2021a) and Araujo et al. (2021) in comparison to Ryu et al. (2019). This experience illustrates that small errors in the estimation of a single Lipschitz constant layer can lead to major errors on the Lipschitz bound of the overall network and that acute precision is crucial. Moreover, methods using PI such as (Singla and Feizi, 2021a) have a huge standard deviation in this task which is problematic for deep networks whereas ours and Sedghi et al. (2019) have standard deviations that remain small. Our method offers the same precision quality as Sedghi et al. (2019) but significantly faster.

These results demonstrate that precise spectral norm calculations for individual layers are critical for obtaining tighter Lipschitz bounds. This precision not only improves robustness guarantees but also reduces variability, making the bounds more reliable for deep networks. However, the PUB is still a loose bound, especially when we compare it to empirical local Lipschitz computed with gradient ascent.

4.2 Controlling the Lipschitz constant of convolutional layers through regularization

Regularization of spectral norms of convolutional layers has been studied in previous works (Yoshida and Miyato, 2017; Miyato et al., 2018; Gouk et al., 2021). Wang et al. (2020) regularized singular values by enforcing orthogonalization, offering a computationally cheap alternative. Achour et al. (2022) further studied this orthogonalization technique. The goal of this experiment is to assess how different bounds control the Lipschitz constant $\mathrm{L}(f^{(l)})$ of each convolution layer along the training process. To perform this control, a target Lipschitz constant γ is set for all convolutions. The loss optimized during training becomes: $\mathcal{L}_{\text{train}} + \mu_{\text{reg}} \sum_{l=1}^{L} \mathcal{L}_{\text{reg}}(f^{(l)})$, with

$$\mathcal{L}_{\text{reg}}(f^{(l)}) = \max\left\{\sigma_{1\,\text{method}}(f^{(l)}),\, \gamma\right\} \tag{4.1}$$

This regularization loss ensures that the Lipschitz constant of the convolutional layer remains close to the target value γ, by penalizing the maximum between the spectral norm and γ. The function $x \mapsto \max\{x, \gamma\}$ is continuous but not differentiable at $x = \gamma$; however, it admits well-defined subgradients. See Figure 4.1 for an illustration of

this regularization loss. We use ResNet18 architecture (He et al., 2016), trained on

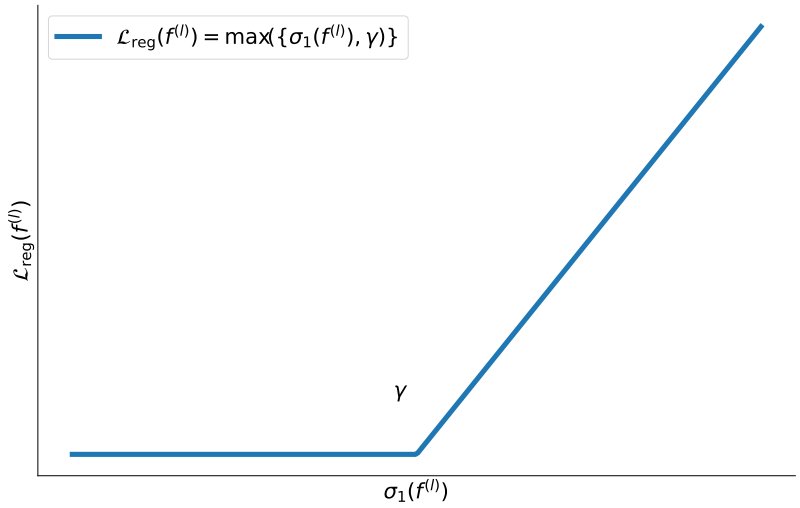


Fig. 4.1.: Illustration of the loss $\mathcal{L}_{\text{reg}}$ in function of the largest singular value $\sigma_{1\,\text{method}}(f^{(l)})$.

the CIFAR-10 dataset for 200 epochs, and with a batch size of 256. We use SGD with a momentum of 0.9 and an initial learning rate of 0.1 with a cosine annealing schedule. The baseline is trained without regularization. We compare different types of regularization: Lipschitz regularization using various methods, all with $\mu_{\text{reg}} = 10^{-1}$, and standard weight decay (WD) applied to the convolutional filters with $\mu_{\text{reg}} = 5 \times 10^{-3}$. This weight decay coefficient was chosen to ensure that the spectral norms of the layers remain approximately bounded around the target value, thus serving as a baseline for comparison. We only compare methods under the circular padding hypothesis and use (Sedghi et al., 2019) as the reference since it provides an exact value. We take 6 iterations for Gram iteration in our method, 10 for power iteration in (Singla and Feizi, 2021a), and 10 samples in (Araujo et al., 2021) to have similar training times. For our method norm2_circ, see Algorithm 5, we implement an explicit backward differentiation using Equation 3.6 to speed up the gradient computation and reduce the memory footprint.

Figure 4.2 presents results on Lipschitz constant control of convolutional layers for ResNet18. The goal here is to have a Dirac distribution at target value $\gamma = 1$, *i.e.* we want each convolutional layer to have a spectral norm of 1. We observe that the more precise a method is, the more well-controlled the resulting distribution of the convolutional layers' spectral norm is after training. Our regularization method matches each layer's target spectral norm of 1. Weight decay gives gross estimations and (Araujo et al., 2021) overestimates spectral norms and thus results in over-constrained Lipschitz constant for some layers. (Singla and Feizi, 2021a) produces a maximum spectral of 1.1 norm which is above target γ but overall Lipschitz constant is centered around the target with dispersion. This suggests that the convolution's spectral norm is only loosely constrained, and that the differentiation of our bound

behaves as expected. Detailed spectral norm histograms over epochs for each Lipschitz regularization method are presented in Figure **??**, showing that all spectral norms are below the target Lipschitz constant at epoch 120 for our method and accounts for fast convergence in Lipschitz regularization in comparison to other methods.

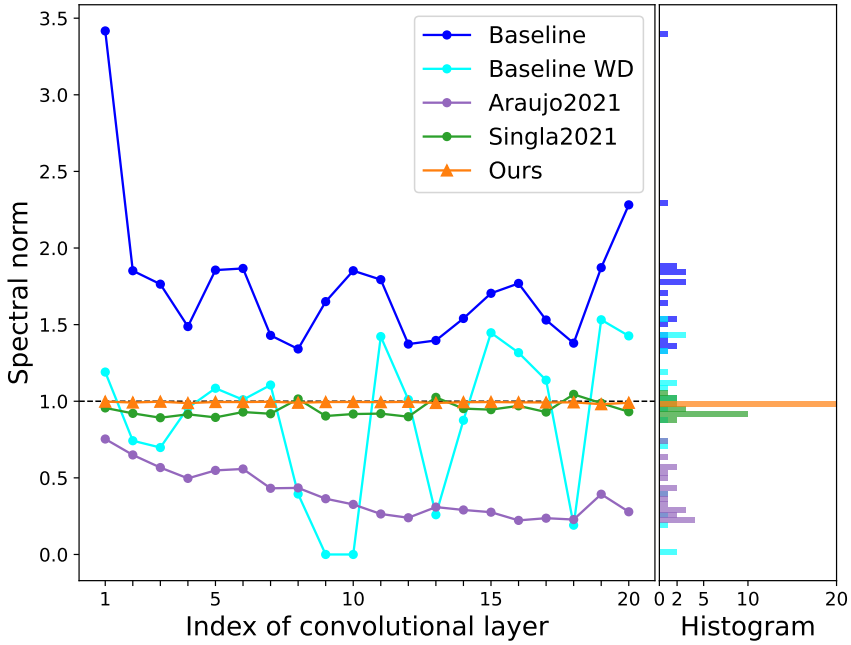


Fig. 4.2.: Plot and histogram of spectral norms for each convolution layer in ResNet18 at the end of training on CIFAR-10, for different regularization methods. We observe that the histogram of spectral norms regularized with our method is all at the target Lipschitz constant, meanwhile, other methods' histograms are scattered.

We provide a study on the impact of Lipschitz regularization on the generalization performance of the network in Section 7.1, using the same experimental setup as in this section.

4.3 Local Lipschitz property of neural networks

Neural networks used are theoretically Lipschitz continuous, but estimating a meaningful Lipschitz constant is challenging. A common approach to obtain a cheap upper bound is to use the $\mathrm{PUB}(f)$. While this bound is easy to compute, it can be extremely loose for deeper networks, as the actual local Lipschitz behavior may be much smaller than this product. 1-Lipschitz networks control the overall Lipschitz constant by design, using a product of upper bounds across layers and making it close to true Lipschitz. However, this global upper bound can be arbitrarily loose compared to the actual local Lipschitz constant around a given input point $\mathbf{x}$, particularly for standard architectures like ResNet-110. Table 4.2 illustrates the significant

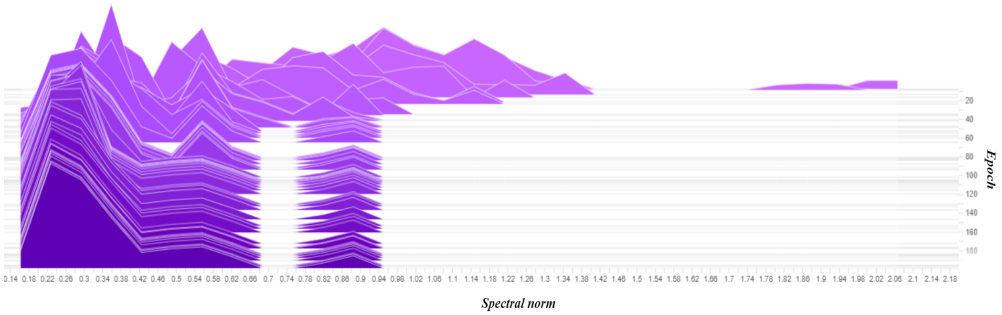


Fig. 4.3.: Spectral norm histograms in the function of epoch for training on CIFAR10 with regularization using Araujo et al. (2021).

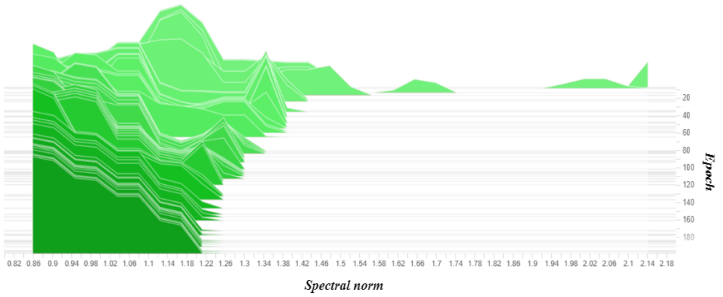


Fig. 4.4.: Spectral norm histograms in the function of epoch for training on CIFAR10 with regularization using Singla and Feizi (2021a).

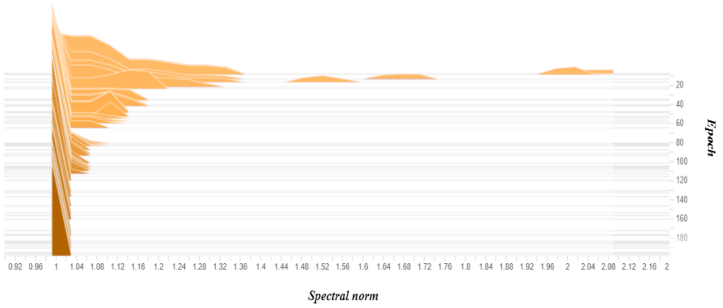


Fig. 4.5.: Spectral norm histograms in the function of epoch for training on CIFAR10 with regularization using our method.

Tab. 4.2.: Comparison of local Lipschitz constant estimation (Local Lip.) and product upper bound (PUB) for different models and noise levels (σ) used during training.

Model	**Local Lipschitz**	**PUB**
LiResNet ($\sigma = 0.00$)	22	37
LiResNet ($\sigma = 0.12$)	11	13
LiResNet ($\sigma = 0.25$)	7.9	9
LiResNet ($\sigma = 0.50$)	4.8	6
ResNet-110 ($\sigma = 0.00$)	235	2.34×10^{10}
ResNet-110 ($\sigma = 0.12$)	27	1.03×10^{12}
ResNet-110 ($\sigma = 0.25$)	25	3.19×10^{12}
ResNet-110 ($\sigma = 0.50$)	19	9.71×10^{9}
ResNet-110 ($\sigma = 1.0$)	3.8	1.32×10^{11}

discrepancy between local Lipschitz estimates given by empirical Lipschitz estimation (Section 2.1.1), and the PUB, especially for ResNet-110, where the global bound becomes impractically loose. This gap underscores the limitations of relying on PUB for certification, motivating the need for research focused on deriving certified bounds for local Lipschitz constants. To elucidate why the empirical certified radius often exceeds the theoretical one, we estimate the local Lipschitz constant $L(\mathbf{x}, \mathcal{B})$ in a neighborhood of x of radius $\epsilon = 0.15$ called $\mathcal{B}$, as detailed in Equation 2.3. However, because these local Lipschitz estimates do not constitute rigorous global upper bounds, they cannot be directly employed for formal certification.

Training with Gaussian noise injection helps to reduce the empirical local Lipschitz constant, making the network smoother and more robust. For example, increasing the noise level σ significantly decreases the empirical local Lipschitz constant, as shown in Table 4.2. This local reduction in sensitivity can be understood through the lens of function smoothing: as shown by Cohen et al. (2019), training with input noise effectively trains a smoothed version of the network, obtained by the Weierstrass transform $\tilde{f}$ of the original function f. Indeed, by Jensen's inequality,

$$\mathcal{L}\big(\tilde{f}(\mathbf{x}), y\big) \leq \mathbb{E}\big(\mathcal{L}\big(f(\mathbf{x}+\delta), y\big)\big),$$

illustrating that minimizing the expected noisy loss indirectly minimizes the loss of the smoothed function. Moreover, explicit bounds on the Lipschitz constant of $\tilde{f}$ are known, showing that smoothing inherently reduces sensitivity to input perturbations.

These results highlight two complementary approaches: (1) designing 1-Lipschitz blocks using the Product Upper Bound to explicitly enforce global control of the Lipschitz constant, and (2) leveraging the Weierstrass transform by training with noise injection to induce smoother, more robust networks. Together, these strategies provide a principled foundation for enhancing stability and robustness in neural networks.

4.4 Training stability with Lipschitz regularization

In this section, we investigate the use of Lipschitz regularization to enhance training stability in deep neural networks. We explore different regularization strategies, including Lipschitz regularization applied to all layers and we try to reduce the number of normalization layers such as batch normalization (Ioffe and Szegedy, 2015).

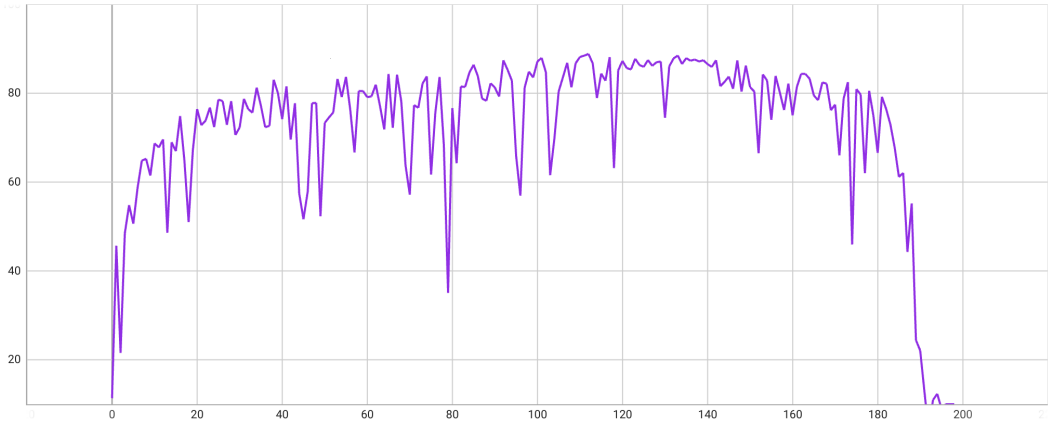


Fig. 4.6.: Classification accuracies (y-axis) for every epcoh (x-axis) for ResNet-18 models trained with Lipschitz regularization applied to all convolutional layers and normalization layers. Training is very unstable.

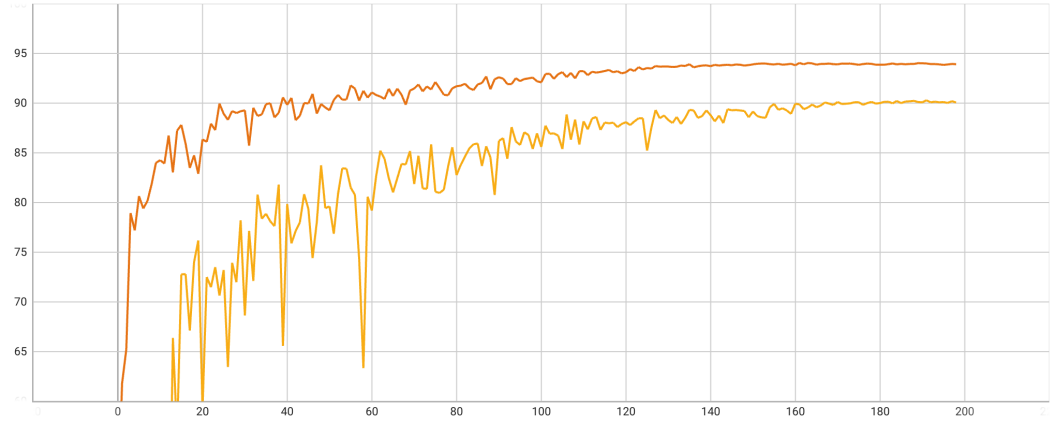


Fig. 4.7.: Comparison of classification accuracies (y-axis) w.r.t epochs (x-axis) for ResNet-18 models trained with Lipschitz regularization (yellow) applied to all layers (All) and a single Batch Normalization layer (OneBatchNorm) vs baseline (orange).

Training with Lipschitz-Regularized layers and batch normalization We train ResNet-18 models with Lipschitz regularization applied to all convolutional, dense, and batch normalization layers, ensuring Lipschitzness. However, training under these constraints proves highly unstable, as shown in Figure 4.6. Batch normalization normalizes activations using statistics computed over the batch, which inherently introduces scale variations. When constrained by Lipschitz regularization, these fluctuations can become unpredictable, potentially causing unstable optimization dynamics. In the following, we try to remove batch normalization layers and replace them with Lipschitz regularization on linear layers.

Training with Lipschitz-Regularized layers and reduced batch normalization Training deep neural networks without batch normalization or other normalization techniques is often unstable due to vanishing and exploding gradients (Ioffe and Szegedy, 2015; Ba et al., 2016). As depth increases, activations and gradients

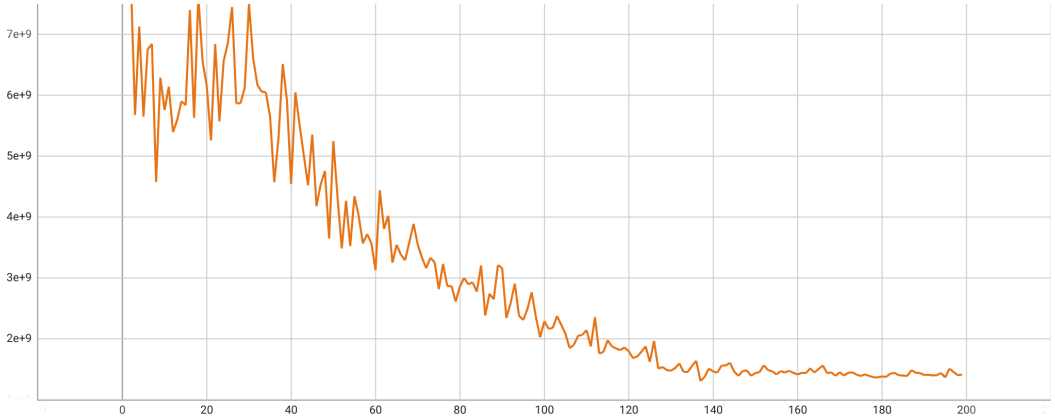


Fig. 4.8.: Evolution of the PUB (y-axis) over training epochs (x-axis) for ResNet-18 models with Lipschitz regularization applied to convolutional/dense layers and batch normalization after each convolutional layer (Baseline).

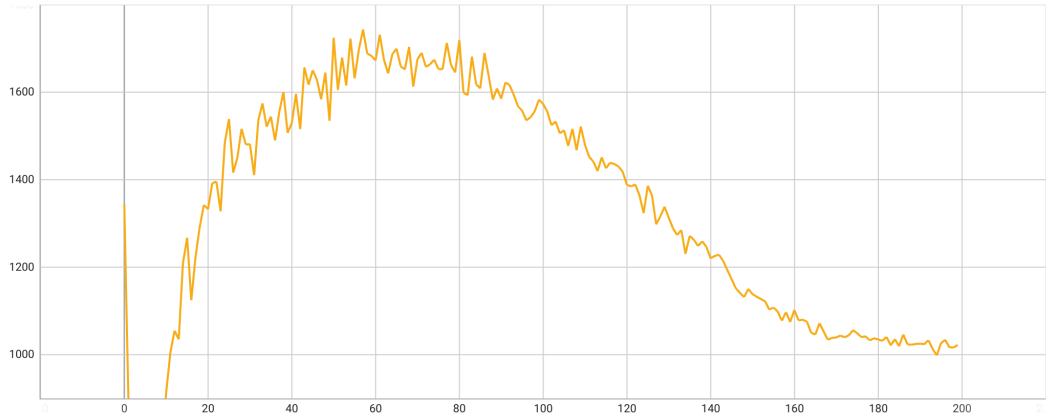


Fig. 4.9.: Evolution of the PUB (y-axis) over training epochs (x-axis) for ResNet-18 models with Lipschitz regularization applied to convolutional/dense layers and a single batch normalization layer after the last layer (OneBatchNorm).

can become poorly scaled, leading to slow convergence or divergence. Batch normalization mitigates these issues by normalizing activations across mini-batches, improving gradient flow, and enabling higher learning rates (Santurkar et al., 2018). However, it also introduces a dependence on batch size, which can be problematic in scenarios such as training on memory-limited edge devices, distributed training with asynchronous updates, or when batch statistics become unreliable due to domain shifts or small dataset sizes. Most CNN architectures incorporate batch normalization after each convolutional layer (He et al., 2016). In such cases, Lipschitz regularization provides a promising alternative, as it stabilizes gradients without relying on batch-dependent normalization. By explicitly constraining the Lipschitz constant of each layer, this regularization method prevents gradients from exploding.

We investigate the impact of Lipschitz regularization on ResNet-18 models under two different normalization strategies: a standard batch normalization setup (Baseline)

and a configuration with a single batch normalization layer (OneBatchNorm). In the (Baseline setup), Lipschitz regularization is applied to all convolutional, dense, and batch normalization layers, following the standard ResNet-18 architecture where batch normalization is placed after each convolutional layer. This enforces strict Lipschitz constraints but leads to significant instability. In contrast, (OneBatchNorm) enforces Lipschitz regularization on the convolutional and dense layers, and structurally replaces all intermediate batch normalization layers by a single batch normalization layer placed just before the final dense (logit) layer. This design aims to preserve the global Lipschitz constraints while still benefiting from normalization, but avoids the instability that multiple batch normalization layers can introduce under Lipschitz control. Figure 4.7 compares classification accuracies, showing that (OneBatchNorm) incurs an accuracy drop of approximately 3% compared to (Baseline). However, as shown in Figures 4.8 and 4.9, the evolution of the Lipschitz upper bound (PUB) differs significantly. In (OneBatchNorm), PUB stabilizes around ≈ 1000. In (Baseline), PUB grows uncontrollably, reaching $\approx 10^9$, indicating severe instability.

While (OneBatchNorm) reduces instability, the accuracy drop and the difficulty in obtaining a tight Lipschitz constant suggest the need for alternative architectures with better gradient norm preservation. Although Lipschitz constraints prevent gradient explosion, they do not inherently address gradient vanishing. This experiment highlights the importance of balancing Lipschitz regularization with architectural choices to ensure stable training and accurate control of the Lipschitz constant.

4.5 Rescaling layers for Lipschitz networks

Unlike the previous sections, which focus on controlling the Lipschitz constant through explicit regularization during training, this section addresses a complementary strategy that enforces Lipschitz constraints directly by design. We build upon existing Lipschitz layers such as the Spectral Lipschitz Layer (SLL), Convex Potential Layer (CPL), Almost Orthogonal Layer (AOL), and Spectral Normalization (SN) by introducing a new approach: spectral rescaling (SR). Spectral Normalization, as introduced by Miyato et al. (2018), divides all singular values of a weight matrix by its largest singular value, ensuring the spectral norm is exactly one. However, this approach squashes smaller singular values towards zero, resulting in an ill-conditioned matrix. The Almost Orthogonal Layer (AOL), proposed by Prach and Lampert (2022), mitigates this issue by applying a diagonal rescaling matrix $\mathbf{R}$ such that the spectral norm of $\mathbf{WR}$ is bounded by one. While this improves the condition number compared to Spectral Normalization, the rescaling is often overly conservative, with the spectral norm $\|\mathbf{WR}\|_2$ significantly lower than one.

To address these limitations, we propose spectral rescaling (SR), a tight rescaling method that interpolates between Spectral Normalization and AOL. Our method converges to a spectral norm of one while maintaining a good condition number, offering improved stability and precision. The rescaling is achieved by generalizing the Gram matrix $\mathbf{W}^\top\mathbf{W}$ to its t-th iterate, providing a more flexible and effective approach to constructing Lipschitz layers. We note the t-th Gram iterate matrix, $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)\star}\mathbf{W}^{(t)}$ and $\mathbf{W}^{(0)} = \mathbf{W}$, is computed using algorithms described in Chapter 3.

Theorem 4.5.1 (Spectral rescaling for matrices)**.**
For any $\mathbf{W} \in \mathbb{R}^{n\times d}$, integer $t \geq 1$, define $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)\top}\mathbf{W}^{(t)}$, with $\mathbf{W}^{(0)} = \mathbf{W}$. We define the spectral rescaling $\mathbf{R}^{(t)}$ as the diagonal matrix with $\mathbf{R}^{(t)}_{ii} = \left(\sum_j \left|\mathbf{W}^{(t+1)}\right|_{ij}\right)^{-2^{-(t+1)}}$ if the expression in the brackets is non zero, or $\mathbf{R}_{ii} = 0$ otherwise.

Then $\sigma_1(\mathbf{W}\mathbf{R}^{(t)}) \underset{t\to+\infty}{\longrightarrow} 1$, with super geometric convergence and the iterates upper bound the limit $\sigma_1(\mathbf{W}\mathbf{R}^{(t)}) \leq 1$.

See proof in Appendix B.2. Using previous Theorem 4.5.1 and Theorem 1 of (Araujo et al., 2023), we can design the following two 1-Lipschitz layers.

Corollary 4.5.2 (Spectral rescaling for 1-Lipschitz layers)**.**
Under the notations of Theorem 4.5.1, let $\mathbf{Q} = \mathrm{diag}(q_i)$ with $q_i > 0$, and fix $t \geq 0$. Define the spectral rescaling matrix $\mathbf{R}^{(t)}$ by

$$\mathbf{R}^{(t)}_{ii} = \begin{cases} \left(\sum_j \left|\mathbf{W}^{(t)}\right|_{ij} \frac{q_i}{q_j}\right)^{-2^{-t}}, & \text{if the sum is nonzero,} \\ 0, & \text{otherwise.} \end{cases}$$

Then, for any $t \geq 0$, the following mappings are 1-Lipschitz:

- *the affine layer $\mathbf{x} \mapsto \mathbf{W}\mathbf{R}^{(t)}\mathbf{x} + \mathbf{b}$,*
- *the residual layer $\mathbf{x} \mapsto \mathbf{x} - 2\mathbf{W}(\mathbf{R}^{(t)})^2\,\rho(\mathbf{W}^\top\mathbf{x} + \mathbf{b})$,*

where $\mathbf{b} \in \mathbb{R}^d$ is a bias vector, $\mathbf{W}$ is the weight matrix, and ρ is any 1-Lipschitz activation (e.g., ReLU, $\tanh$, $\mathrm{sigmoid}$).

Spectral rescaling is differentiable so training is possible through the rescaled layer as a whole. The introduction of the diagonal matrix $\mathbf{Q} = \mathrm{diag}(q_i)$ adds flexibility to the spectral rescaling by enabling anisotropic control over the activation dimensions.

When $\mathbf{Q}$ is learned jointly with the model parameters, it enhances the expressivity of the rescaling operation while preserving 1-Lipschitzness through the construction in Corollary 4.5.2. Each learned rescaling coefficient $q_i > 0$ can adaptively emphasize or attenuate certain directions, allowing the network to better align with the data geometry under the Lipschitz constraint. Previous Corollary 4.5.2 can be applied to convolutional layers using Algorithm 6 to compute the rescaling.

Note that for $t = 1$, SR is equivalent to AOL (Prach and Lampert, 2022) (see Equation 2.5). As $t \to \infty$ the SR is more similar to SN, indeed the stable rank $\|\mathbf{W}^{(t)}\|_{\mathrm{F}}{}^2/\|\mathbf{W}^{(t)}\|_2^2$ of the Gram iterate matrix $\mathbf{W}^{(t)}$, is closer and closer to one. The Gram iterate for large iteration number t is numerically a rank one matrix and the rescaling is equivalent to spectral normalization. Therefore, our method is an interpolation between the AOL rescaling and the SN rescaling, the number of iterations t is the cursor which allows us to select at which degree the rescaling is more AOL or SN. Note that SN used on the SLL block of (Araujo et al., 2023) retrieves the CPL block of (Meunier et al., 2022).

One main advantage of our method over SN done with power iteration is that it provides a guarantee that the layer is 1-Lipschitz as the rescaling is upper bounding the singular values. As we are rescaling, the Lipschitz constant is always lower than 1, see Figure 4.10 the right figure depicts the convergence of SR to a Lipschitz constant of 1 as iteration increases. We see empirically that power iteration (with 100 iterations) can be a loose lower bound on spectral norm as depicted in Figure 3.11. Moreover, SR is deterministic and converges super geometrically, whereas SN with power iteration is random and converges linearly.

We observe that the ratio of singular values $\frac{\sigma_1}{\sigma_i}$ is systematically lower for SR compared to AOL and SN (Figure 4.10, left). This suggests that the singular values are more evenly distributed, meaning that transformations induced by SR are more isotropic than those of AOL and SN. This effect reduces average directional distortions in feature space, potentially leading to more stable representations.

4.6 Results on regularity of the Weierstrass transform

In this section, we present new results on the regularity properties of the Weierstrass transform when the underlying network is Lipschitz continuous. This is particularly relevant for networks designed to be Lipschitz by construction where we would perform on top Weierstrass transform to smooth the network. Also, as we see in the previous section 4.3 the Lipschitz constant of vanilla networks such as ResNet

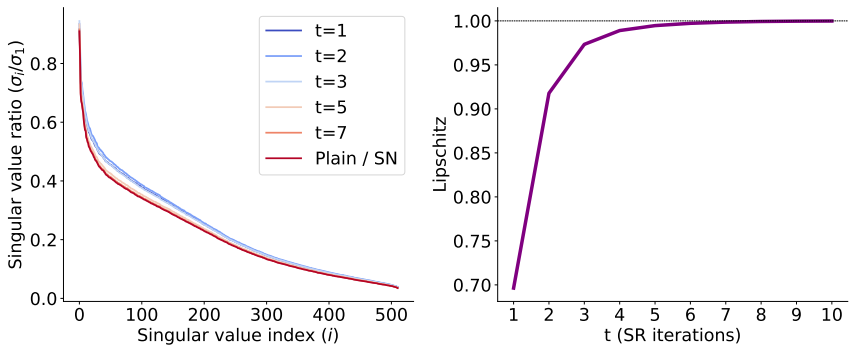


Fig. 4.10.: On the left, the ratio of $\frac{\sigma_i}{\sigma_1}$ for a dense layer from ResNet18 (512×1000) trained on the ImageNet-1k dataset with different rescaling methods. On the right, the convergence of SR to a 1-Lipschitz layer is depicted.

is overestimated by the product of upper bounds and the true Lipschitz constant is probably much smaller. Incorporating this piece of information can help to derive tighter bounds on the Lipschitz constant of the smoothed network.

In the Lemma 2.1.5 of Salman et al. (2019), the Lipschitz constant of the Weierstrass transform is bounded by the Lipschitz constant of the smoothed function f and the standard deviation σ of the Gaussian kernel used for smoothing. Lemma 2.1.6 states that the Lipschitz constant of the Weierstrass transform $\widetilde{f} = f * \phi_\sigma$ is bounded by the Lipschitz constant function f. Note that in each case the two quantities σ and $\mathrm{L}(f)$ do not interact in the bound. We derive enhanced bounds on the Lipschitz constant of $\widetilde{f}$ with the additional assumption that f is Lipschitz continuous. Using the same notation and assumption as previous lemmas we have the following result:

Theorem 4.6.1 (Lipschitz bound for the Weierstrass transform with based Lipschitz continuity)**.**
Suppose that $f : \mathbb{R}^d \to [0,1]$ is Lipschitz continuous, then

$$\mathrm{L}(\widetilde{f}) \leq \mathrm{L}(f)\,\mathrm{erf}\left(\frac{1}{2^{\frac{3}{2}}\mathrm{L}(f)\sigma}\right). \tag{4.2}$$

See proof in Appendix B.3. We can also derive the following bound which depends only on σ and $\mathrm{L}(f)$ independently:

Corollary 4.6.2.
(Lipschitz Bounds on the Weierstrass transform) Suppose that $f : \mathbb{R}^d \to [0,1]$ is Lipschitz continuous, then

$$\mathrm{L}(\widetilde{f}) \leq \frac{1}{\sqrt{2\pi\sigma^2}} \tag{4.3}$$

and also

$$\mathrm{L}(\widetilde{f}) \leq \mathrm{L}(f)\ . \tag{4.4}$$

The smoothed function $\widetilde{f}$ is at least as regular as the original function f, also it is noteworthy that Equation 4.3 enhances the bound on $\mathrm{L}(\widetilde{f})$ originally derived in Lemma 2.1.5 of Salman et al. (2019) by a factor of 2. This refinement on the bound was possible by supposing Lipschitz continuity on the function f. Note that its Lipschitz constant can be arbitrarily high, so this assumption is quite light: the Lipschitz constant does not play into the derived bound. In the proof we demonstrate that the bound is optimal as it exists a function f such that the bound is an equality. These improved bounds can be seamlessly incorporated into subsequent works, such as (Pautov et al., 2022; Franco et al., 2023; Chen et al., 2024a).

We observe that the Weierstrass transform and Lipschitz continuity exhibit a synergistic effect on the Lipschitz constant of the smoothed function $\tilde{f}$. In particular, there exists an intermediate regime, characterized by a specific relationship between σ and $\mathrm{L}(f)$, where these effects combine to reduce the Lipschitz constant of $\tilde{f}$ more than either mechanism would individually. To formalize this, we define the gap function

$$\Delta(\sigma) = \min\{\mathrm{L}(f),\ \frac{1}{\sqrt{2\pi}\,\sigma}\}\ -\ \mathrm{L}(f)\,\mathrm{erf}\Big(\frac{1}{2^{3/2}\mathrm{L}(f)\sigma}\Big),$$

which quantifies the improvement provided by considering the refined Weierstrass bound over the independent Lipschitz and smoothing bounds.

Proposition 4.6.3.

Let $f : \mathbb{R}^d \to [0,1]$ be a Lipschitz continuous function. The optimal smoothing parameter that maximizes this gap, i.e.

$$\sigma^* = \arg\max_{\sigma>0} \Delta(\sigma),$$

is given explicitly by

$$\sigma^* = \frac{1}{\mathrm{L}(f)\sqrt{2\pi}} \quad\Longrightarrow\quad \mathrm{L}(\tilde{f})\ \leq\ \mathrm{erf}\big(\tfrac{\sqrt{\pi}}{2}\big)\,\mathrm{L}(f)\ \approx\ 0.79\,\mathrm{L}(f).$$

See proof in Appendix B.4. For this choice of σ^*, we also have that the randomized smoothing bound exactly meets the deterministic Lipschitz bound, since

$$\frac{1}{\sqrt{2\pi}\,\sigma^*} = \mathrm{L}(f).$$

This reveals a principled trade-off: by selecting σ in relation to the intrinsic Lipschitz constant $\mathrm{L}(f)$, or by designing $\mathrm{L}(f)$ to match a target smoothing scale, we maximize the synergistic reduction captured by $\Delta(\sigma)$. Consequently, this combination can reduce the certified Lipschitz bound by up to 21%. This provides a practical guideline: given either a desired smoothing level or an inherent Lipschitz constraint, one can tune the complementary parameter to maximize the benefits of Lipschitz control and Weierstrass smoothing captured through $\Delta(\sigma)$.

To illustrate the impact of this result, we consider the example of the smoothed variant of binary probit regression where Gaussian noise is injected into the input during inference. This technique enhances the robustness of the model by averaging predictions over perturbations of the input, mitigating sensitivity to adversarial noise. Logistic regression is a linear model for binary classification that predicts the probability of an input $\mathbf{x} \in \mathbb{R}^d$ belonging to the positive class. It models the relationship between the input features and the output class using the sigmoid function. The prediction is given by:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}},$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and $b \in \mathbb{R}$ is the bias. This output represents the probability of the positive class $y = 1$, while $1 - f(\mathbf{x})$ corresponds to the probability of the negative class $y = 0$. The sigmoid function $s(z) = \frac{1}{1+e^{-z}}$ ensures that the output is bounded between 0 and 1, making it interpretable as a probability. Logistic regression is typically trained by minimizing the cross-entropy loss, which measures the difference between the predicted probabilities and the ground truth labels.

Example 4.6.4 (Example of smoothed binary probit regression)**.**
Consider a linear function $x \mapsto \mathbf{w}^\top \mathbf{x} + b$ with $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. The smoothed probit regression model is defined as:

$$\tilde{f}(\mathbf{x}) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \mathbf{I})}[s(\mathbf{w}^\top (\mathbf{x} + \delta) + b)],$$

where $s(z)$ is an approximation of the sigmoid function, expressed as:

$$s(z) = \Phi(\lambda z),$$

with Φ representing the Gaussian cumulative distribution function. Choosing $\lambda = \sqrt{\frac{\pi}{9}}$ or $\lambda = \sqrt{\frac{\pi}{5.35}}$ provides a close approximation to the standard sigmoid $s(z) = \frac{1}{1+e^{-z}}$. These values minimize approximation error, ensuring that the smoothed sigmoid closely matches the standard sigmoid across various inputs.

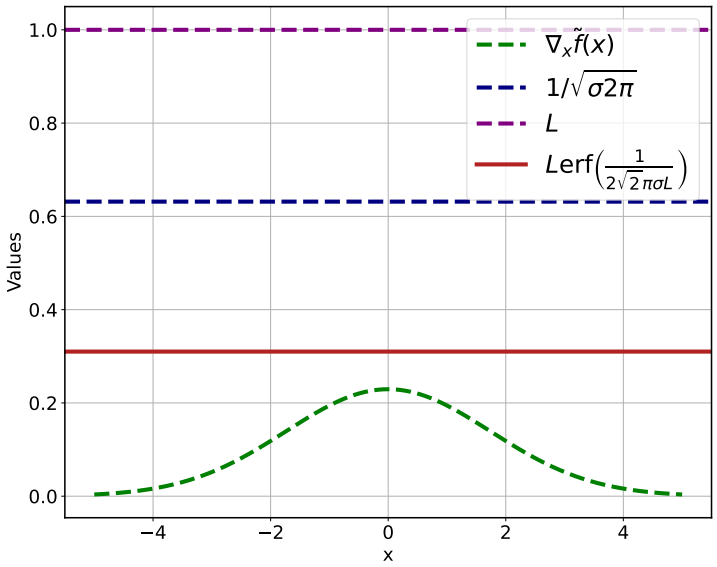


Fig. 4.11.: Plot of the gradient and bounds of function $\widetilde{f}_k$ for a smoothed probit regression model with $d = 1$ and $\sigma = 0.2$ and $\mathrm{L}(f) = 1$.

The closed-form solution for the smoothed output, following [Eq. 4.152]Bishop (2006), is:

$$\tilde{f}(\mathbf{x}) = \Phi\left(\frac{\lambda(\mathbf{w}^\top \mathbf{x} + b)}{(1 + \lambda^2\sigma^2\|\mathbf{w}\|^2)^{\frac{1}{2}}}\right).$$

This formulation demonstrates how smoothing with Gaussian noise yields an analytical solution, providing robustness against perturbations and adversarial inputs.

Figure 4.11 shows the gradient and bounds of the smoothed function $\widetilde{f}_k$ for a smoothed probit regression model, see Example 4.6.4, with $\mathrm{L}(f) = 1$ and associated $\sigma^\star = \frac{1}{\sqrt{2\pi}}$. The bounds are computed using the Lipschitz constant of the original function f and the RS one with standard deviation σ of the Gaussian noise. We observe that our bound $\mathrm{L}(\widetilde{f})$ is tighter around the gradient of the smoothed function, demonstrating the effectiveness of the bound using the Lipschitz constant mixed with smoothing standard deviation in controlling the regularity of the smoothed function $\tilde{f}$.

We derive a similar result for the Lemma 2.1.7 of (Salman et al., 2019) by incorporating Lipschitz continuity assumptions on the network. This result provides a tighter bound on the Lipschitz constant of the quantile-composed smoothed network. The following theorem provides a key result for deriving certified radii under the assumption of Lipschitz continuity on the function f for the randomized smoothing framework.

Theorem 4.6.5 (Weierstrass transform with local Lipschitz continuity)**.** *The function $\Phi^{-1} \circ \tilde{f}$ is locally Lipschitz continuous within the ball $B(\mathbf{x}, \epsilon) = \{\mathbf{x}' \in \mathbb{R}^d : \|\mathbf{x}' - \mathbf{x}\|_2 \leq \epsilon\}$, with Lipschitz constant:*

$$\mathrm{L}\left(\Phi^{-1} \circ \tilde{f}, B(\mathbf{x}, \epsilon)\right) \leq \mathrm{L}(f) \sup_{\mathbf{x}' \epsilon B(\mathbf{x}, \epsilon)} \left\{ \frac{\Phi_\sigma\left(s_0(\mathbf{x}') + \frac{1}{\mathrm{L}(f)}\right) - \Phi_\sigma\left(s_0(\mathbf{x}')\right)}{\phi(\Phi^{-1}(\tilde{f}(\mathbf{x}')))} \right\}$$

where Φ_σ is the cumulative distribution function of the normal distribution with standard deviation σ, ϕ is the standard Gaussian density function, and $s_0(\mathbf{x}')$ is determined by solving:

$$\tilde{f}(\mathbf{x}') = 1 - \mathrm{L}(f) \int_{s_0(\mathbf{x}')}^{s_0(\mathbf{x}') + \frac{1}{\mathrm{L}(f)}} \Phi_\sigma(s)\, ds \ .$$

See proof in Appendix B.5. Incorporating the Lipschitz constant of f gives a local Lipschitz constant for the quantile-composed smoothed classifier. This result provides a tighter bound on the Lipschitz constant of the quantile-composed smoothed network. Here the choice of an optimal $\sigma^\star$ is more complex than in the previous case, as it depends on the Lipschitz constant is local and it depends on $\mathbf{x}$ and radius ϵ. Same as before we can compute the expectation under Gaussian noise injected

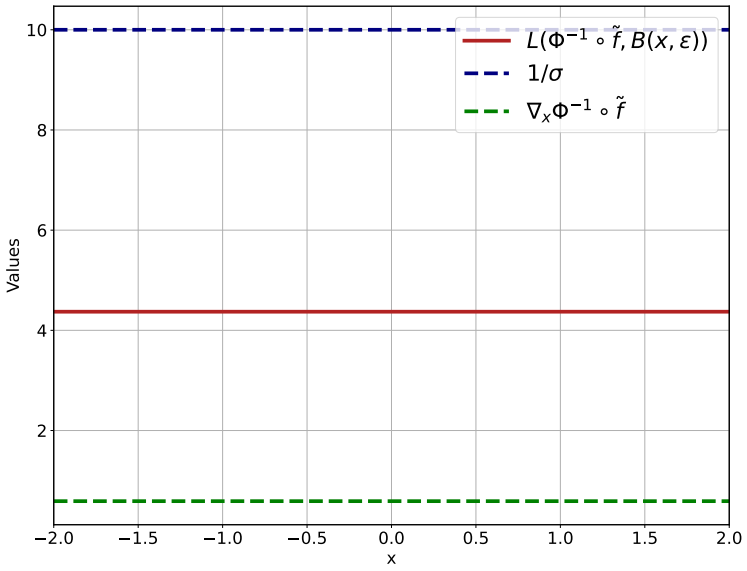


Fig. 4.12.: Plot of the gradient and bounds of function $\Phi^{-1} \circ \widetilde{f_k}$ for a smoothed probit regression model with $d = 1$ and $\sigma = 0.1$ and $\mathrm{L}(f) = 1$.

over input exactly for a probit regression model, see Example 4.6.4. Figure 4.12 shows the gradient and bounds of the smoothed function $\Phi^{-1} \circ \widetilde{f_k}$ for a smoothed probit regression model with $d = 1$, $\mathrm{L}(f) = 1$ and associated $\sigma = 0.1$.

4.7 Conclusion

This chapter presented an in-depth exploration of Lipschitz networks, focusing on methods to estimate, regularize, and design neural architectures with controlled Lipschitz constants. We introduced the Product Upper Bound (PUB) as an efficient technique for estimating the Lipschitz constant of convolutional networks, demonstrating its accuracy and scalability.

To further constrain the Lipschitz constant during training, we investigated regularization strategies applied to convolutional layers, showing that precise spectral control enhances network stability and reduces variability. We also proposed spectral rescaling (SR), a novel method that interpolates between existing normalization techniques, providing tighter bounds and better conditioning for deep networks.

Additionally, we explored the theoretical properties of the Weierstrass transform, highlighting its role in smoothing neural networks. We examined the interaction between Lipschitz continuity and the Weierstrass transform, demonstrating how their combined effect can lead to tighter bounds on the Lipschitz constant of smoothed networks.

While our work has focused on estimating, regularizing, and designing Lipschitz networks, another perspective highlights the critical role of loss functions in shaping the accuracy-robustness trade-off (Béthune et al., 2022). This suggests that beyond architectural constraints, optimization choices and loss design play a fundamental role in fully leveraging the benefits of Lipschitz-constrained models. A deeper understanding of these interactions could bridge the gap between theoretical guarantees and practical performance, paving the way for models that are both provably robust and highly expressive. The techniques developed in this chapter contribute to this broader effort by addressing key aspects of network stability and expressivity, with direct implications for robustness against adversarial perturbations. In the next chapter, we apply these tools developed in the current chapter to certified robustness, demonstrating how bounding the Lipschitz constant provides formal guarantees on adversarial resilience and connects theoretical stability measures to practical security concerns in deep learning.

5 Robustness through Lipschitz networks

Contents

Deep learning models are highly susceptible to adversarial attacks, where imperceptible perturbations to input data can drastically alter model predictions (Szegedy et al., 2013). This vulnerability raises significant concerns for the deployment of neural networks in safety-critical applications such as autonomous driving, medical diagnosis, and security-sensitive systems. Addressing these concerns requires robust models capable of certifying their predictions against adversarial perturbations (Katz et al., 2017). Lipschitz continuity provides a theoretical foundation for quantifying and enforcing robustness by controlling how much a model's output can change in response to input variations. Constraining the Lipschitz constant has been shown to enhance robustness against adversarial perturbations (Tsuzuku et al., 2018). However, enforcing these constraints in deep architectures remains challenging due to computational difficulties in estimating and maintaining a tight Lipschitz bound. Certified robustness techniques, such as randomized smoothing, offer a probabilistic guarantee against adversarial attacks by constructing smoothed classifiers that are inherently more stable. However, a critical limitation of current certification methods is that the obtained certified robustness is significantly lower than the empirical robustness observed under adversarial attacks such as projected gradient descent (PGD) (Cohen et al., 2019; Croce et al., 2021). This gap between theoretical guarantees and empirical observations limits the practical effectiveness of certification techniques and raises questions about their applicability in real-world scenarios.

In this chapter, we evaluate the certified robustness of Lipschitz networks under both deterministic and non-deterministic settings. We first assess robustness using

Spectrally Rescaled (SR) layers, then extend the analysis to randomized smoothing with the Weierstrass transform, deriving new certified radius bounds incorporating Lipschitz constraints. This analysis aims to bridge the gap between theoretical robustness guarantees and practical improvements, highlighting the interplay between Lipschitz regularization and smoothing techniques.

5.1 Spectrally rescaled layers for certified robustness

Lipschitz networks improve robustness by explicitly constraining the Lipschitz constant of each layer, ensuring that the overall Lipschitz bound remains controlled through the product upper bound (PUB). A notable property of Lipschitz networks is their stability during training: a low Lipschitz constant mitigates gradient explosion and stabilizes optimization dynamics (Kodali et al., 2018). Indeed, unlike standard networks, Lipschitz architectures inherently reduce the need for batch normalization, layer normalization, or gradient clipping, as their controlled gradients naturally promote stable convergence. However, enforcing a Lipschitz constraint can also introduce challenges: the contractive nature of such layers may lead to vanishing gradients, thereby limiting the effective depth of trainable networks. An important property to counteract this issue is norm preservation, which helps mitigate gradient attenuation and is crucial for training deep Lipschitz networks (Anil et al., 2019).

Residual connections are essential for addressing this issue, as they facilitate stable gradient propagation and mitigate vanishing gradients. As demonstrated by He et al. (2016) with ResNet, residual layers enable the training of deeper networks by preserving gradient flow across layers. This property is particularly valuable for Lipschitz networks, where the controlled weight magnitudes further ensure stability. Additionally, residual architectures have been shown to boost performance, as highlighted by Araujo et al. (2023).

In this work, we employ SLL layers which is a Lipschitz residual layer, which requires rescaling to enforce the 1-Lipschitz constraint. The baseline method utilizes Almost Orthogonal Layers (AOL) Prach and Lampert, 2022 for rescaling, while we introduce Spectral Rescaling (SR) to improve both robustness and accuracy. As detailed in Corollary 4.5.2, the rescaling matrix $\mathbf{R}$ is defined by:

$$\mathbf{R}_{ii}^{(t)} = \left(\sum_j \left| \mathbf{W}^{(t)} \right|_{ij} \frac{\mathbf{q}_i}{\mathbf{q}_j} \right)^{-2^{-t}} \quad \text{if the sum is non-zero,} \quad \text{otherwise} \quad \mathbf{R}_{ii}^{(t)} = 0.$$

Using the Corollary 4.5.2, for any layer $1 \leq l \leq L$, the following residual structure is 1-Lipschitz:

$$f^{(l)}(\mathbf{x}) = \mathbf{x} - 2\mathbf{W}(\mathbf{R}^{(t)})^2 \rho(\mathbf{W}^\top \mathbf{x} + \mathbf{b}),$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ the bias vector, and $\rho = \mathrm{ReLU}$ is the activation function.

As each layer for $1 \leq l \leq L$ is 1-Lipschitz, the overall network is also 1-Lipschitz, thus

$$\mathrm{L}(f) = \prod_{l=1}^{L} \mathrm{L}(f^{(l)}) = 1 \ .$$

We can then use the certified radius bound from Tsuzuku et al. (2018), see Equation 2.10, to compute the certified robustness of the network and the certified accuracy on the validation set.

Experimental setup and results. We evaluate the robustness of Lipschitz networks designed with SLL layers by comparing AOL and SR rescaling methods. Following the experimental setup of Araujo et al. (2023), we conduct experiments on CIFAR-10 and CIFAR-100 datasets. Training is performed over 200 epochs with a batch size of 256, using standard data augmentation techniques. The networks are optimized with Adam (Kingma and Ba, 2017), with a learning rate of 0.01 and parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$, without weight decay. A piecewise triangular learning rate scheduler is applied to adjust the learning rate during training. The loss function follows the CrossEntropy formulation used by Prach and Lampert (2022), with a temperature of 0.25 and an offset value of $\frac{3}{2}\sqrt{2}$. This ensures consistent evaluation across architectures and datasets, providing a reliable comparison between rescaling methods. To evaluate the scalability of SLL SR based Lipschitz networks, we consider two architectures: small and medium. The architectures are defined as: small (S): 20 convolutional layers, 7 dense layers, and medium (M): 30 convolutional layers, 10 dense layers.

Certified performance. Tables 5.1 and 5.2 summarize natural and certified accuracies (see Definition 2.2.3) across perturbation levels ϵ, averaged over three training runs. SR consistently outperforms AOL across all architectures and datasets, showcasing enhanced scalability and robustness. The tighter Lipschitz bounds achieved through SR layers improve network stability and certifiability.

Spectral rescaling consistently enhances certified robustness and network stability across datasets and architectures, offering a scalable solution for certifiably robust networks.

Tab. 5.1.: Certified accuracy on CIFAR-10 for different perturbation levels ϵ, comparing AOL and SR rescaling methods for small (S) and medium (M) architectures. Mean values are reported over 3 runs, standard deviation is omitted for brevity and is bounded by 0.12 for all configurations.

Rescaling	Accuracy	Certified accuracy (ϵ)			
		0.141	0.283	0.423	1
AOL (S)	71.06	62.78	53.67	45.37	19.18
SR (S)	**72.44**	**63.49**	**54.66**	**46.01**	**19.62**
AOL (M)	72.41	63.72	54.48	46.38	19.92
SR (M)	**73.38**	**64.64**	**55.43**	**46.79**	**20.18**

Tab. 5.2.: Certified accuracy on CIFAR-100 for different perturbation levels ϵ, comparing AOL and SR rescaling methods for small (S) and medium (M) architectures. Mean values are reported over 3 runs, standard deviation is omitted for brevity and is bounded by 0.13 for all configurations.

Rescaling	Accuracy	Certified accuracy (ϵ)			
		0.141	0.283	0.423	1
AOL (S)	46.05	35.13	26.72	20.49	0.073
SR (S)	**46.62**	**35.46**	**27.42**	**21.14**	**0.075**
AOL (M)	46.50	35.81	27.39	21.21	7.78
SR (M)	**47.50**	**36.50**	**28.54**	**21.89**	**8.13**

5.2 Certified robustness with randomized smoothing of Lipschitz networks

In the background sections, we discussed the two main approaches for certifying robustness: deterministic, see Section 2.2.1, and probabilistic, see Section 2.2.2. The deterministic approach, as proposed by Tsuzuku et al. (2018), relies on a Lipschitz network to provide a deterministic certificate. This method is computationally efficient but may be overly conservative, when the Lipschitz constant is bounded by the PUB, leading to a potential drop in performance. The probabilistic approach, as proposed by Cohen et al. (2019), smooths the soft classifier $F = \tau \circ f$ (see Equation 2.11), to create a soft smoothed classifier $\widetilde{F}$, which is estimated with Monte-Carlo integration to provide a probabilistic certificate with risk α. This method is more computationally expensive but offers a more accurate and flexible certification as it accounts for the architecture as a whole.

Recall that $F = \tau \circ f$, where f is the Lipschitz network, τ maps to the simplex, and $F_k : \mathbb{R}^d \to [0,1]$ denotes the k-th component for $c = |\mathcal{Y}|$ classes. The smoothed classifier $\widetilde{F}$ is defined by:

$$\widetilde{F}_k(\mathbf{x}) = \int_{\mathbb{R}^d} F_k(\mathbf{x}+\delta)\phi_\sigma(\delta)d\delta, \tag{5.1}$$

where $\phi_\sigma(\delta) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{-\|\delta\|^2/(2\sigma^2)}$ is the Gaussian kernel with standard deviation σ.

5.2.1 Lipschitz interpretation of randomized smoothing

Randomized smoothing, as described by Salman et al. (2019), can be interpreted through the Weierstrass transform, where smoothing is achieved by convolving the classifier output with a Gaussian kernel:

$$\widetilde{F}(\mathbf{x}) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\left[F(\mathbf{x}+\delta)\right] = \left(F * \phi_\sigma\right)(\mathbf{x}).$$

This highlights the relationship between randomized smoothing and Lipschitz continuity, particularly the smoothness of the element-wise score from classifier $\widetilde{F}_k$, for $1 \le k \le c$. Leveraging this connection, we can derive two kinds of certified radius for $\widetilde{F}$.

First certified radius bound. Using this interpretation, developed in Salman et al., 2019, we can use the expression from Section 2.2.1. To compute the certified radius, we first apply the coordinate-wise bound $\mathrm{R}_{\text{coord}}$ from Equation 2.9. By plugging in the Lipschitz constant of the smoothed classifier, derived from Lemma 2.1.5, from Salman et al. (2019), for all $1 \le k \le c$:

$$\mathrm{L}(\widetilde{F}_k) \le \sqrt{\frac{2}{\pi\sigma^2}}\ ,$$

we get the following certified radius bound:

$$\mathrm{R}_{\text{coord}}(\widetilde{F}, \mathbf{x}, y) \le \frac{\sqrt{\pi\sigma^2}}{2\sqrt{2}}\mathrm{M}(\widetilde{F}(\mathbf{x}), y)\ .$$

To obtain a better bound, we can also use the refined bound derived in Corollary 4.6.2,

$$\mathrm{L}(\widetilde{F}_k) \le \frac{1}{\sqrt{2\pi\sigma^2}}\ ,$$

which gives a tighter bound for the Lipschitz constant of the smoothed classifier:

$$\mathrm{R}_{\mathrm{coord}}(\widetilde{F}, \mathbf{x}, y) \leq \sqrt{\frac{\pi\sigma^2}{2}}\mathrm{M}(\widetilde{F}(\mathbf{x}), y) \ . \tag{5.2}$$

which is larger by a factor 2.

To derive Lipschitzness for the smoothed classifier $\widetilde{F}$, the outputs of F must be bounded. Typically, for a softmax classifier, these outputs lie in the interval $[0, 1]$. Consequently, the margin $\mathrm{M}(\widetilde{F}(\mathbf{x}), y)$ is bounded by 1. This results in a maximum certified radius of

$$\mathrm{R}_{\mathrm{coord}}(\widetilde{F}, \mathbf{x}, y) \leq \sqrt{\frac{\pi\sigma^2}{2}}, \tag{5.3}$$

which is considerably lower than the empirical robustness upper bound obtained through adaptive attacks, as reported by Chen et al. (2024b). This certified radius is derived from the "weak law" of randomized smoothing, which assumes that the maximum Lipschitz condition holds along the entire perturbation path (Chen et al., 2024a). In contrast, the "strong law" of randomized smoothing, as presented in Lemma 2.1.7 (Salman et al., 2019), allows for non-constant Lipschitzness for $\widetilde{F}$. This leads to a tighter and more accurate robust radius, with the potential for the upper bound of the certified radius to be unbounded, as it makes intervene $\Phi^{-1} \circ \widetilde{F}$ which is not bounded. This observation motivates the exploration of an alternative certified radius based on the Lipschitz constant of the composition $\mathrm{L}(\Phi^{-1} \circ \widetilde{F})$.

Second certified radius bound. Same as for the first radius, we can plug the Lipschitz constant of the smoothed classifier, derived from Lemma 2.1.7 in $\mathrm{R}_{\mathrm{coord}}$.

$$\mathrm{R}_{\mathrm{coord}}(\Phi^{-1} \circ \widetilde{F}, \mathbf{x}, y) = \frac{\mathrm{M}(\Phi^{-1} \circ \widetilde{F}(\mathbf{x}), y)}{2\mathrm{L}(\Phi^{-1} \circ \widetilde{F}_k)} \ .$$

Rewriting it, with $\mathrm{L}(\Phi^{-1} \circ \widetilde{F}_k) = \frac{1}{\sigma}$, we get the multi-class certified radius bound:

$$\mathrm{R}_{\mathrm{mult}}(\widetilde{F}(\mathbf{x}), y) = \frac{\sigma}{2}\left(\Phi^{-1}(\widetilde{F}_y(\mathbf{x})) - \max_{k \neq y} \Phi^{-1}(\widetilde{F}_k(\mathbf{x}))\right) \ , \tag{5.4}$$

which is similar to the radius derived in Cohen et al. (2019), see Theorem 2.2.5, for the choice $\underline{\mathbf{p}}_{i_1} = \mathbf{p}_{i_1}$ and $\overline{\mathbf{p}}_{i_2} = \mathbf{p}_{i_2}$, with $\mathbf{p} = \widetilde{F}(\mathbf{x})$.

5.2.2 Randomized smoothing on Lipschitz networks

In the background section, we derive new Lipschitz bound on the smoothed classifiers $\widetilde{F}$ and $\Phi^{-1} \circ \widetilde{F}$, in particular in the case where underlying classifier $F = f \circ \tau$ is

Lipschitz by design, thus and $\mathrm{L}(\widetilde{F})$ is known. This requires that τ to be a mapping on $[0,1]^c$ such as $\tau = \mathrm{softmax}$ or $\tau(x) = \frac{1}{1+e^{-x}}$ for instance. Following the same interpretation of the previous section we can examine the cross-effect of randomized smoothing and Lipschitz continuity on the Lipschitz constant of the smoothed classifier $\widetilde{F}$.

First certified radius bound. As in the previous section, using the Lipschitz constant of the smoothed classifier element-wise, from Theorem 4.6.1, for all $1 \leq k \leq c$:

$$\mathrm{L}(\widetilde{F}_k) \leq \mathrm{L}(F_k)\,\mathrm{erf}\left(\frac{1}{2^{\frac{3}{2}}\mathrm{L}(F_k)\sigma}\right).$$

We can derive the certified radius for $\widetilde{F}$:

$$\mathrm{R}_{\mathrm{coord}}(\widetilde{F},\mathbf{x},y) \geq \underbrace{\mathrm{erf}\left(\frac{1}{2^{\frac{3}{2}}\mathrm{L}(F_k)\sigma}\right)^{-1}}_{\text{randomized smoothing}} \underbrace{\frac{\mathrm{M}(\widetilde{F}(\mathbf{x}),y)}{2\mathrm{L}(F_k)}}_{\text{Lipschitz deterministic}}. \tag{5.5}$$

We focus on an intermediate regime defined by a specific σ and $\mathrm{L}(F)$, where these effects interact in a manner that is mutually beneficial, exceeding the individual impacts of randomized smoothing or Lipschitz continuity alone. Using the Proposition 4.6.3, optimal $\sigma^\star$ is given by

$$\sigma^\star = \frac{1}{\sqrt{\pi}\mathrm{L}(F_k)}.$$

For this choice, we obtain a certificate, see Equation. 5.5, for the smoothed classifier $\widetilde{F}$ that is up to 26% larger than the maximum certification given by randomized smoothing, see Equation 5.2, or Lipschitz continuity alone, see Equation 2.9. In the following we explain why this is the case. We assume the same margins $m = \mathrm{M}(F(x),y) = \mathrm{M}(\tilde{F}(x),y)$ for the deterministic network and smoothed network. In the deterministic case, the certified radius is given by $\mathrm{R}_{\mathrm{det}} = \frac{m}{L}$. For randomized smoothing with an optimal choice $\sigma^\star = \frac{1}{L\sqrt{2\pi}}$, we have

$$\mathrm{Lip}(\tilde{F}) \leq L\,\mathrm{erf}\big(\tfrac{\sqrt{\pi}}{2}\big),$$

which leads to a certified radius

$$\mathrm{R}_{\mathrm{rs}} = \frac{m}{\mathrm{Lip}(\tilde{f})} \geq \frac{1}{\mathrm{erf}(\frac{\sqrt{\pi}}{2})}\frac{m}{L}.$$

This implies a maximum relative gain over the deterministic certificate of

$$\frac{\mathrm{R}_{\mathrm{rs}}}{\mathrm{R}_{\mathrm{det}}} \geq \frac{1}{\mathrm{erf}(\frac{\sqrt{\pi}}{2})} \approx \frac{1}{0.79} \approx 1.27,$$

which corresponds to a certified radius about 27% larger.

Those bounds on the certified radius are using $\mathrm{R}_{\text{coord}}$, so they don't take into account the overall Lipschitz constant $\mathrm{L}(\widetilde{F})$ and the radius $\mathrm{R}_{\text{global}}$ from Tsuzuku et al. (2018). To derive a bound for $\mathrm{R}_{\text{global}}$, we can leverage the fact that F takes values in the probability simplex Δ^{c-1}, to derive a Lipschitz bound for the smoothed classifier $\widetilde{F}$.

We can adapt the Theorem 4.6.1 to the case where τ is a mapping on the probability simplex Δ^{c-1}.

Corollary 5.2.1 (Global Lipschitz constant of the smoothed classifier)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}^c$ be a Lipschitz function, $\tau : \mathbb{R}^c \to \Delta^{c-1}$ be a 1-Lipschitz mapping on the simplex and the soft classifier $F = \tau \circ f$. Then the* soft smoothed classifier *$\tilde{F}$, is Lipschitz continuous with:*

$$\mathrm{L}(\widetilde{F}) \leq \mathrm{L}(F)\,\mathrm{erf}\left(\frac{1}{2\mathrm{L}(F)\sigma}\right) \leq \min\left\{\frac{1}{\sqrt{\pi\sigma^2}}, \mathrm{L}(F)\right\} .$$

Using this corollary we get the following certified radius bound for the smoothed classifier $\widetilde{F}$:

$$\mathrm{R}_{\text{global}}(\widetilde{F}, \mathbf{x}, y) \geq \underbrace{\mathrm{erf}\left(\frac{1}{2\mathrm{L}(F)\sigma}\right)^{-1}}_{\text{randomized smoothing}} \quad \underbrace{\frac{\mathrm{M}(\widetilde{F}(\mathbf{x}), y)}{\sqrt{2}\mathrm{L}(F)}}_{\text{Lipschitz deterministic}} . \tag{5.6}$$

Second certified radius bound. For the second radius, which uses the $\Phi^{-1} \circ \widetilde{F}$ we can use the Theorem 4.6.5 to derive a bound for the Lipschitz constant of $\Phi^{-1} \circ \widetilde{F}$. The difficulty is that we have a local Lipschitz constant, given by

$$\mathrm{L}\left(\Phi^{-1} \circ \tilde{F}_k, B(\mathbf{x}, \epsilon)\right) \leq \mathrm{L}(F_k) \sup_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} \left\{ \frac{\Phi_\sigma\left(s_0(\mathbf{x}') + \dfrac{1}{\mathrm{L}(F_k)}\right) - \Phi_\sigma\left(s_0(\mathbf{x}')\right)}{\phi(\Phi^{-1}(\tilde{F}_k(\mathbf{x}')))} \right\} ,$$

and $s_0(\mathbf{x}')$ is determined by solving:

$$\tilde{F}_k(\mathbf{x}') = 1 - \mathrm{L}(F_k) \int_{s_0(\mathbf{x}')}^{s_0(\mathbf{x}') + \frac{1}{\mathrm{L}(F_k)}} \Phi_\sigma(s)\, ds .$$

Thus the certified radius $\mathrm{R}_{\text{coord}}$ is also local, let say certification is required for a budget ϵ, and we have to test if the certified radius is superior to ϵ:

$$\mathrm{R}_{\text{multLip}}(\Phi^{-1} \circ \widetilde{F}, \mathbf{x}, y) \geq \frac{1}{2}\left(\frac{\widetilde{F}_y(\mathbf{x})}{\mathrm{L}(\Phi^{-1} \circ \widetilde{F}_k, B(\mathbf{x}, \epsilon))} - \max_{k \neq y} \frac{\widetilde{F}_k(\mathbf{x})}{\mathrm{L}(\Phi^{-1} \circ \widetilde{F}_k, B(\mathbf{x}, \epsilon))}\right) \geq \epsilon \ . \tag{5.7}$$

A summary of the bounds on the Lipschitz constant of the smoothed classifier is presented in Table 5.3.

Tab. 5.3.: Summary of bounds on the Lipschitz constant of the smoothed classifier. In this table $b = s_0(\mathbf{x}')$ and $a = b + \dfrac{1}{\mathrm{L}(F_k)}$

Case	Bound expression	Reference
$\mathrm{L}(\widetilde{F}_k)$ element-wise	$\mathrm{L}(\widetilde{F}_k) \leq \sqrt{\frac{2}{\pi\sigma^2}}$	Salman et al. (2019), Lemma 2.1.5
$\mathrm{L}(\widetilde{F}_k)$ element-wise (refined)	$\mathrm{L}(\widetilde{F}_k) \leq \frac{1}{\sqrt{2\pi\sigma^2}}$	Corollary 4.6.2
$\mathrm{L}(\widetilde{F}_k)$ for Lipschitz F_k	$\mathrm{L}(\widetilde{F}_k) \leq \mathrm{L}(F_k)\ \mathrm{erf}\left(\frac{1}{2^{\frac{3}{2}}\mathrm{L}(F_k)\sigma}\right) \leq \min\left\{\frac{1}{\sqrt{2\pi\sigma^2}}, \mathrm{L}(F_k)\right\}$	Theorem 4.6.1
$\mathrm{L}(\widetilde{F})$ global	$\mathrm{L}(\widetilde{F}) \leq \mathrm{L}(F)\ \mathrm{erf}\left(\frac{1}{2\mathrm{L}(F)\sigma}\right) \leq \min\left\{\frac{1}{\sqrt{\pi\sigma^2}}, \mathrm{L}(F)\right\}$	Corollary 5.2.1
$\mathrm{L}(\Phi^{-1} \circ \widetilde{F})$ global	$\mathrm{L}(\Phi^{-1} \circ \widetilde{F}) \leq \frac{1}{\sigma}$	Standard RS theory, Cohen et al. (2019)
$\mathrm{L}(\Phi^{-1} \circ \widetilde{F})$ local	$\mathrm{L}(\Phi^{-1} \circ \widetilde{F}_k, B(\mathbf{x}, \epsilon)) \leq \mathrm{L}(F_k) \sup_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} \frac{\Phi_\sigma(a) - \Phi_\sigma(b)}{\phi(\Phi^{-1}(\widetilde{F}_k(\mathbf{x}')))}$	Thm 4.6.5

5.2.3 Generalized simplex mapping

In classification tasks, model outputs are typically mapped onto the standard probability simplex $\Delta^{c-1} \subset \mathbb{R}^c$, where entries are non-negative and sum to 1, forming a valid class distribution. In the randomized smoothing (RS) framework, the output has traditionally been treated as a probability distribution due to the probabilistic nature of the method. Certified radii are derived by analyzing the probability of class scores under Gaussian noise, relying on the assumption that network outputs represent valid probabilities. While this probabilistic perspective has shaped the development of RS, strict adherence to the simplex is not essential for deriving Lipschitz bounds or applying smoothing techniques like the Weierstrass transform. For smoothed networks to exhibit Lipschitz continuity, it suffices for the underlying function to be bounded, allowing greater flexibility in the choice of mappings while still enabling the derivation of meaningful certified radii.

Relaxing the mass constraint leads to the generalized scaled simplex ${\Delta^{c-1}}_r$, where entries sum to an arbitrary value $r > 0$. Adjusting r influences the scale of the

distribution, modifying model margins and Lipschitz properties, thus providing a tool to balance robustness and stability. Indeed for a simplex of mass r, the margins are bounded by r, and this can unlock a new certified radius beyond the mass-1 simplex, see Equation 5.3.

Mappings such as $\mathrm{softmax}$ are often employed to transform logits onto the standard simplex due to their smooth, 1-Lipschitz properties. However, $\mathrm{softmax}$ can compress margins by producing dense distributions. To address this, Martins and Astudillo, 2016 proposed $\mathrm{sparsemax}$, which projects logits onto Δ^{c-1} while encouraging sparsity, resulting in larger margins and preserving 1-Lipschitz continuity. The $\mathrm{sparsemax}$ projection onto the simplex is defined as:

$$\mathrm{sparsemax}(\mathbf{z}) = \underset{\mathbf{p}\in\Delta^{c-1}}{\arg\min} \|\mathbf{p}-\mathbf{z}\|_2^2,$$

where Δ^{c-1} denotes the probability simplex. The projection onto the simplex performed by *sparsemax* is a classical operation in convex optimization. The specific algorithm used to compute sparsemax is closely related to the algorithm of Held et al. (1974), and later formalized in modern form by Condat (2016) for fast projection onto the simplex or ℓ_1-ball.

Why sparsemax encourages sparsity. This projection leads to sparsity because it sets all components z_i below a certain threshold $\tau(\mathbf{z})$ to zero:

$$\mathrm{sparsemax}_i(\mathbf{z}) = \max\left(z_i - \tau(\mathbf{z}), 0\right),$$

where the threshold $\tau(\mathbf{z})$ is computed such that $\sum_i \mathrm{sparsemax}_i(\mathbf{z}) = 1$ and only the largest components of $\mathbf{z}$ remain positive. Consequently, many entries are exactly zero, producing a sparse output vector.

Building on this, we propose a generalized $\mathrm{sparsemax}$ that maps logits onto the r-simplex ${\Delta^{c-1}}_r$, where the total mass r can differ from the conventional value of 1. This generalized projection onto the scaled simplex Δ_r^{c-1} has already been employed in prior work (see, e.g., Condat, 2016, Algo. 1), and we adopt it here to control the sparsity and margin of the output. Lower values of r lead to sparser, more selective distributions, whereas higher values allow for larger margins between active classes. The generalized $\mathrm{sparsemax}$ is a 1-Lipschitz mapping towards Δ_r^{c-1}, following the same proof as in Laha et al., 2018, Appendix A.5. This mapping is described in Algorithm 7. Recall that $F = \tau \circ f$ where f is the Lipschitz network and τ is a generalized simplex mapping. For $r_1 \leq r_2$, when most of the logit vectors $f(x+\delta_i)$ are bounded by r_1, the mapping τ^{r_2} to the simplex $\Delta_{r_2}^{c-1}$ with 1-Lipschitz simplex mapping is not going to increase the margin associated to vectors $\tau^{r_2}(f(x+\delta_i))$. In

Algorithm 7 generalized_sparsemax($\mathbf{z}$, r)

1: Sort $\mathbf{z}$ in decreasing order $\mathbf{z}_1 \geq \cdots \geq \mathbf{z}_c$
2: Find $\kappa(z)$ such that

$$\kappa(\mathbf{z}) = \max_{k=1\ldots c} \left\{ k \,\middle|\, r + k\mathbf{z}_k > \sum_{j \leq k} \mathbf{z}_j \right\}$$

3: Define

$$\rho(\mathbf{z}) = \frac{\left(\sum_{j \leq \kappa(\mathbf{z})} \mathbf{z}_j\right) - r}{\kappa(\mathbf{z})}$$

4: **return** $\mathbf{p}$ such that $\mathbf{p}_i = \max(\mathbf{z}_i - \rho(\mathbf{z}), 0)$

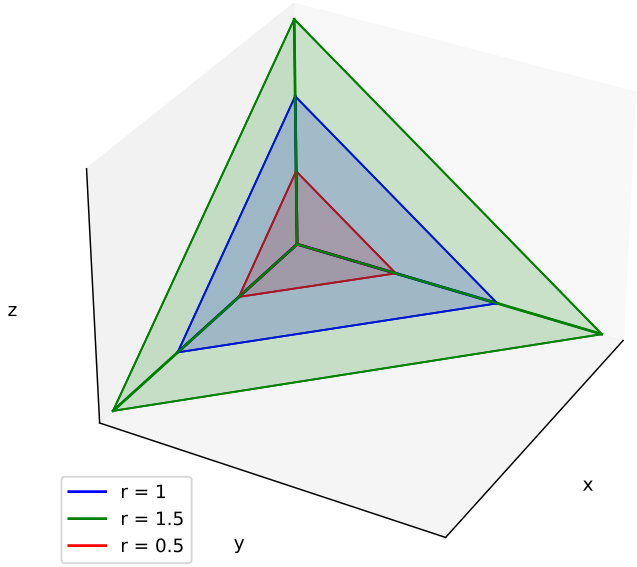


Fig. 5.1.: Illustration of simplexes of different mass $r \in \{0.5, 1, 1.5\}$ for $c = 3$ classes.

this case, it is better to map to the simplex $\Delta^{c-1}_{r_1}$ of lower mass and enjoy a tighter Lipschitz constant on $\tilde{F}$ or $\Phi^{-1} \circ \tilde{F}_k$. Conversely, when $r_2 \geq r_1$, one can benefit from larger margins on $\tau^{r_2}(f(x + \delta_i))$ in comparison to margins on $\tau^{r_1}(f(x + \delta_i))$. In Figure 5.1, we can see simplexes of different sizes, one point on a simplex gives a margin at most equal to the mass of the simplex.

Theorem results that establish Lipschitz bounds for functions $F : \mathbb{R}^c \to \mathbb{R}^c$ can extend naturally to $F : \mathbb{R}^d \to {\Delta^{c-1}}_r$, optimizing certified radii without restrictive simplex constraints. We get the following corollary for the Lipschitz constant of the *smoothed classifier* $\tilde{F}$:

Corollary 5.2.2.
Let $F : \mathbb{R}^d \to \mathbb{R}^c$ be a L-Lipschitz function, $\tau : \mathbb{R}^c \to {\Delta^{c-1}}_r$ be a 1-Lipschitz mapping on

Tab. 5.4.: Certified accuracy on CIFAR-10 for different levels of perturbation ϵ, for RS, Lipschitz deterministic, and ours. The risk is taken as $\alpha = 1\text{e}{-3}$ and the number of samples $n = 10^4$.

Methods	**Certified accuracy (ϵ)**						**Average time (s)**
	0.14	0.19	0.25	0.28	0.42	0.5	
Lipschitz deterministic	40	33.57	27.18	24.59	13.65	9.15	**0.004**
Randomized smoothing	47.9	31.99	28.17	27.86	6.42	0.0	0.9
RS with new bound	**52.56**	**46.17**	**39.09**	**35.08**	**21.9**	**13.53**	0.9

the r-simplex and the soft classifier $F = \tau \circ F$. Then the soft smoothed classifier $\tilde{F}$*, is Lipschitz continuous with:*

$$\mathrm{L}(\widetilde{F}) = \mathrm{L}(F)\,\mathrm{erf}\left(\frac{r}{2\mathrm{L}(F)\sigma}\right) \leq \min\left\{\frac{r}{\sqrt{\pi\sigma^2}}, \mathrm{L}(F)\right\}. \tag{5.8}$$

This flexibility in tuning r makes the r-simplex and the generalized $\mathrm{sparsemax}$ valuable tools, particularly for balancing the trade-off between variance and margin. For instance, a smaller r reduces the Lipschitz constant further, enhancing robustness, while a larger r increases the margins between classes, see Corollary 5.2.2. This approach extends traditional probabilistic frameworks in randomized smoothing by offering a spectrum of robustness profiles that go beyond the conventional mass-1 simplex. We can also show that optimal $\sigma^\star$ is given by $\sigma^\star = \frac{r}{\sqrt{\pi}\mathrm{L}(f)}$.

5.2.4 Experiments

Impact of Lipschitz constant on the first radius. To illustrate the gain of having a Lipschitz bound of the smoothed classifier $\widetilde{F}$ which includes information on the Lipschitz constant of base classifier F, simplex mass r and variance σ^2, we compare certified accuracies on the same by design 5-Lipschitz neural network backbone $\mathrm{Sandwhich\ Small}$ from (Wang and Manchester, 2023), trained with noise injection $\sigma = 0.4$ and using the same certified robust radius $\mathrm{R}_{\mathrm{global}}$ in Equation 2.10.

We choose for the smoothing variance $\sigma^\star = \frac{r}{L(F)\sqrt{\pi}}$ as it maximizes the gain of our new bound, we use a Lipschitz constant of 5 because 1 is too restricitive for the network to be trained with this scale of noise injection. Remark that the variance used for noise injection to train the 5-Lipschitz classifier F is close to $\sigma^\star$ to mitigate a drop in performance on the smoothed classifier $\widetilde{F}$. We consider the three procedures: Lipschitz deterministic using bound on $L(F)$, the RS using bound on $L(\tilde{F})$, and our approach using new bound on $L(\tilde{F})$. For ours and RS, we fix $r = 3$. Results are displayed in Table 5.4. We see that our procedure gives better-certified accuracies

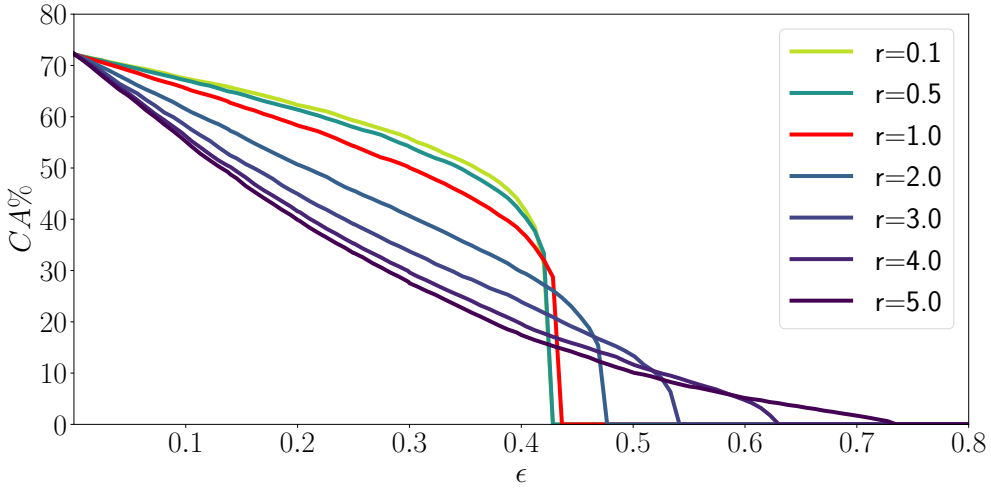


Fig. 5.2.: Certified accuracies (CA in $\%$) with $\mathrm{R}_{\mathrm{coord}}$ in function of levels of perturbation r on CIFAR-10, for different simplex mass r. Number of samples is $n = 10^4$ and risk $\alpha = 1e\text{-}3$. The case $r = 1$ in the color red corresponds to the regular RS probabilistic setting.

than RS and Lipschitz deterministic taken alone, indeed both methodologies provide the same Lipschitz constant for $\tilde{F}$ and F respectively, whereas our method provides an inferior Lipschitz bound on $\mathrm{L}(\tilde{F})$. Note that better results from the random procedure should not be directly construed as an intrinsic superiority over the deterministic one, as the element of randomness introduces variability that must be accounted for in the evaluation and large sampling computational cost. However, it gives a perspective on the performance of the theoretical Lipschitz smoothed classifier $\tilde{F}$.

Impact of simplex mass on the first radius. We note that to reduce the Lipschitz constant, one can not only increase the smoothing noise σ, as is traditionally done in randomized smoothing (RS) but also reduce r, the total mass of the simplex. This introduces a new degree of freedom in the design of smoothed classifiers. In Fig. 5.2, we plot the evolution of certified accuracies for different simplex masses r, using the same settings as described previously. It becomes evident that the classical RS setting, i.e. $r = 1$, is merely one specific instance of a broader range of robustness profiles. By varying the simplex mass r, we can explore and unlock new, potentially larger certified radii, depending on the desired magnitude of the certification radius. This allows for more flexibility in tailoring the robustness of the classifier to meet specific application requirements.

Note that for $\epsilon = 0$ we recover the natural accuracy of the classifier, which is around 70% for the considered network architecture on the CIFAR-10 dataset. It does not depend on the simplex mass r as only the $\arg\max$ operation is performed over the logits and margins are not meaningful in this case.

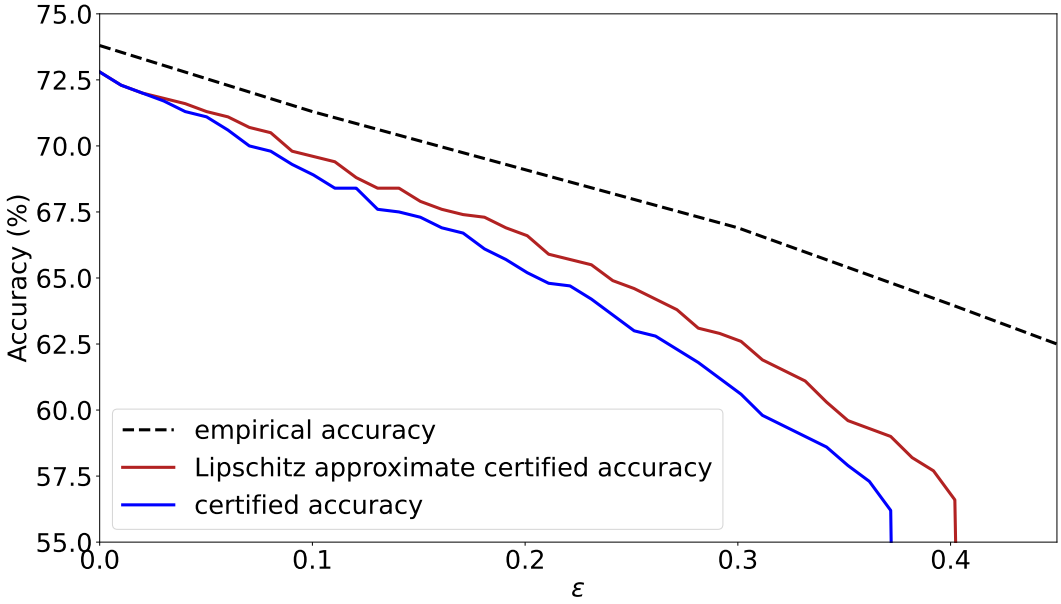


Fig. 5.3.: LiResNet trained with noise injection ($\sigma = 0.5$), on CIFAR-10 dataset. Certified accuracy R_{multi} vs Lipschitz approximate certified accuracy R_{multiLip} vs empirical robust accuracy using projected gradient ascent. Randomized smoothing noise is taken as $\sigma = 0.12$.

Impact of Lipschitz constant on second radius. In the paper of Cohen et al. (2019) the authors point out that the produced certificates were too conservative compared to obtained empirical robustness. This could be due to the fact that the Lipschitz constant of the classifier F is not taken into account in the certified radius. To derive the certified radius R_{multiLip}, we acknowledge that this value is not exactly computable, as it requires evaluating the local Lipschitz constant through a certified procedure. However, to illustrate the potential gains of incorporating such Lipschitz constants into the certified radius under the "strong law" of randomized smoothing (RS), we employ Lipschitz gradient ascent locally (Section 2.1.1) to estimate $\mathrm{L}(\Phi^{-1} \circ \widetilde{F}, B(0, \epsilon))$.

In this approximation, we utilize LiResNet (Hu et al., 2023), a state-of-the-art Lipschitz network for certified robustness, trained on CIFAR-10. The resulting radius R_{multi} underestimates the certified radius R_{multiLip}, demonstrating that the Lipschitz continuity assumption tightens the bound on the true certified radius. We compared the standard certified robustness obtained from the classical RS procedure, which yields the radius R_{multi}, to our new Lipschitz-adjusted radius R_{multiLip}. Both approaches apply the standard Bonferroni correction and the empirical Bernstein confidence bound (Maurer and Pontil, 2009). For empirical evaluation, we conducted assessments using a projected gradient descent (PGD) ℓ_2 attack (Madry et al., 2018) with 40 iterations and a step size of 0.2. This experiment illustrates the potential improvements in certified robustness through the integration of the Lipschitz-adjusted radius.

As shown in Figure 5.3, the certified radius obtained with the Lipschitz continuity assumption, R_{multiLip}, more closely aligns with empirical robust accuracy (i.e., accuracy on the test set under the PGD ℓ_2 attack). This suggests that the inclusion of the Lipschitz assumption not only tightens the certified bound but also produces results that better reflect the actual robustness observed in practice, compared to the standard certified radius R_{multi}. It is important to note that empirical robustness evaluations should ideally involve directly attacking the smoothed classifier, as demonstrated in Salman et al. (2019), or utilize stronger adversarial attacks than PGD. Consequently, the empirical robustness reported here is likely slightly overestimated, as attacking the smoothed classifier is generally more challenging and often results in lower robustness values.

5.3 Conclusion

In this chapter, we explored the robustness properties of Lipschitz networks through deterministic and probabilistic approaches. We demonstrated that spectrally rescaled layers (SR) enhance certified robustness by maintaining 1-Lipschitz constraints. This approach consistently outperformed baseline rescaling methods across multiple datasets and architectures.

We extended our analysis by integrating randomized smoothing (RS) with Lipschitz networks, highlighting the interaction between smoothing noise σ and the Lipschitz constant. By varying the simplex mass r, we introduced a new degree of freedom in certified robustness, unlocking larger certified radii compared to traditional RS methods. This generalized smoothing approach enables the exploration of a broader range of robustness profiles, providing greater flexibility in network design.

Furthermore, we demonstrated the impact of local Lipschitz constants on the second certified radius R_{multiLip}. Through empirical evaluations on CIFAR-10, we showed that incorporating Lipschitz continuity assumptions aligns certified bounds more closely with empirical robust accuracy, outperforming classical RS techniques. This suggests a promising avenue for addressing the gap between certified radii in theory and those observed in practice. By refining the estimation of local Lipschitz constants, this approach holds the potential to further reduce the discrepancy, offering a pathway towards tighter and more realistic robustness guarantees.

Overall, our results indicate that Lipschitz-aware smoothing techniques can significantly enhance the robustness of neural networks. The next chapter's work will focus on controlling the variance and risk in certification induced by the Monte-Carlo integration to estimate the smoothed classifier in randomized smoothing.

6 Risk control and variance reduction in randomized smoothing

Contents

Randomized smoothing (RS) enhances adversarial robustness by averaging predictions over Gaussian-perturbed inputs, providing formal certification of classifier robustness. A key challenge in RS is balancing the tradeoff between margin and variance. While maximizing classification margins improves robustness, it also increases variance, leading to overly conservative robustness estimates. Mono-class certified radii, R_{mono}, often fail under high smoothing variance, as the top-class probability may fall below the certification threshold. Multi-class radii, R_{multi}, mitigate this by considering the top two class probabilities, providing meaningful certification even when the top class confidence is low.

However, introducing R_{multi} necessitates controlling confidence intervals across all class probabilities, which scales with the number of classes. This can result in highly conservative bounds, particularly in datasets with many classes. To address this, we propose the Class Partitioning Method (CPM), which groups less significant classes into meta-classes, reducing the number of confidence intervals that must be

managed. This improves the standard confidence interval obtained with Bonferroni correction, lowering conservativeness and enabling tighter certified radii without increasing computational complexity.

Additionally, we introduce the Lipschitz-Variance-Margin (LVM) framework to reduce variance at the model level. By controlling the Lipschitz constant and applying temperature-scaled simplex mappings, LVM stabilizes smoothed classifiers, further enhancing certified accuracy across various perturbation levels.

6.1 Certification procedures for randomized smoothing

Randomized smoothing (RS) enhances adversarial robustness by averaging predictions over Gaussian perturbations of the input, producing a smoothed classifier.

$$\tilde{F}(\mathbf{x}) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2\mathbf{I})}[F(\mathbf{x}+\delta)].$$

Let $\mathbf{p} = \widetilde{F}(x)$, where $\mathbf{p} \in \Delta^{c-1}$ is the probability vector over c classes, $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_c)$, with $\mathbf{p}_i$ representing the probability of class i. RS provides a certified radius that quantifies robustness to adversarial perturbations, derived from the top two class probabilities $\mathbf{p}_{i_1}$ and $\mathbf{p}_{i_2}$. For a risk level α, the certified radius is computed with a confidence interval for the probabilities $\mathbf{p}_{i_1}$ and $\mathbf{p}_{i_2}$ at level $1-\alpha$:

$$\mathrm{R}_{\mathrm{mult}}(\mathbf{p}) = \frac{\sigma}{2}\left(\Phi^{-1}(\mathbf{p}_{i_1}) - \Phi^{-1}(\mathbf{p}_{i_2})\right) .$$

In practise, a simpler mono-class version is often used: $\mathrm{R}_{\mathrm{mono}}(\mathbf{p}) = \sigma\Phi^{-1}(\mathbf{p}_{i_1}) \leq \mathrm{R}_{\mathrm{mult}}(\mathbf{p})$ as it requires only the top class probability $\mathbf{p}_{i_1}$ and is easier to compute.

6.1.1 Advocating multi-class over mono-class certified radii

When the top-class probability $\mathbf{p}_{i_1}$ is high, the mono-class certified radius $\mathrm{R}_{\mathrm{mono}}$ provides a meaningful robustness guarantee. This is illustrated in Figure 6.1, where the confidence in a single dominant class leads to a non-trivial certified region. However, in binary classification settings—particularly when class i_1 is certified against a superclass that aggregates all remaining classes—the mono-class bound $\mathrm{R}_{\mathrm{mono}}$ becomes ineffective when $\mathbf{p}_{i_1} \leq \frac{1}{2}$, as highlighted by Voráček and Hein (2023) and Voráček (2024). Figure 6.2 demonstrates this limitation: when the predicted top-class probability falls below $\frac{1}{2}$, the mono-class radius drops to zero, indicating a complete lack of certified robustness.

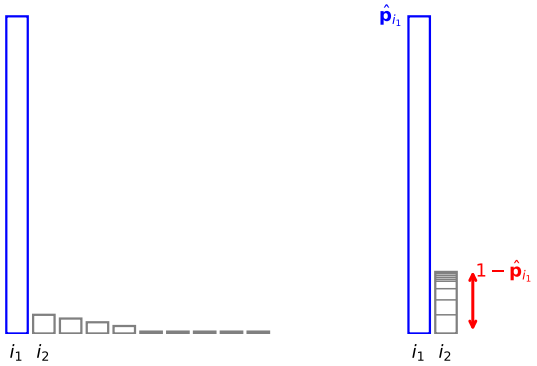


Fig. 6.1.

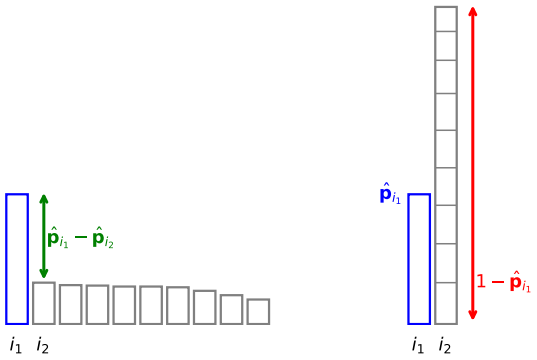


Fig. 6.2.

To address this, the multi-class radius $\mathrm{R}_{\mathrm{mult}}$, which incorporates both $\mathbf{p}_{i_1}$ and the second-highest class probability $\mathbf{p}_{i_2}$, can yield a non-zero certification even when $\mathbf{p}_{i_1} < \frac{1}{2}$. Nonetheless, under high noise regimes (large variance σ^2), this approach also shows limitations. The distribution of $F(x+\delta)$ tends to flatten, approaching a uniform distribution. Consequently, the gap between $\mathrm{R}_{\mathrm{mult}}$ and $\mathrm{R}_{\mathrm{mono}}$ widens, as reported in Table 6.1 and previously observed by Voráček and Hein (2023). In this table n is the number of samples used to estimate the certified radii, α is the risk level, and σ is the smoothing variance.

Tab. 6.1.: Comparison of two certified radii $\mathrm{R}_{\mathrm{mult}}$ and $\mathrm{R}_{\mathrm{mono}}$, and total variation distance to uniform distribution (TVU), for different values of σ. We took a subset of images from the ImageNet test set with $n = 10^4$, $\alpha = 0.001$. If the effect is not visible for $\sigma < 0.25$, we see that as the TVU decreases, the difference between the two radii increases as well.

σ	TVU	$\mathrm{R}_{\mathrm{mult}}$	$\mathrm{R}_{\mathrm{mono}}$	$\mathrm{R}_{\mathrm{mult}} - \mathrm{R}_{\mathrm{mono}}$
0.25	0.998	**5.89**	**5.89**	0.00
0.30	0.996	**4.68**	4.48	0.20
0.35	0.989	**3.68**	3.21	0.47
0.40	0.986	**2.49**	1.97	0.52
0.50	0.976	**1.28**	0.59	0.69
0.60	0.95	**0.56**	0.00	0.56

6.1.2 Confidence intervals for certified radii

To estimate the certified radius, one must rely on approximations since the smoothed classifier cannot be computed directly. MC integration provides this approximation, the probabilities $\mathbf{p}$ are estimated via sampling:

$$\hat{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} F(\mathbf{x} + \delta_i),$$

where $\delta_i \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. An $(1-\alpha)$-exact coverage confidence interval is needed to control the risk α associated with the certified radius accurately.

Definition 6.1.1 (Exact α coverage confidence interval)**.**
Let ξ be a random variable taking values in $\mathbb{R}$. An $(1-\alpha)$-exact coverage confidence interval for a parameter θ is an interval $[\underline{\xi}, \overline{\xi}]$ such that the probability of θ lying within the interval is greater than $1-\alpha$, i.e.,

$$\mathbb{P}(\underline{\xi} \leq \theta \leq \overline{\xi}) \geq 1 - \alpha.$$

Here, α represents the risk, which is the probability that the interval does not contain the true value of θ.

The inherent uncertainty is controlled by concentration inequalities or Clopper-Pearson confidence interval in the case of $\tau = \mathrm{hardmax}$ simplex mapping, they are used to derive. Commonly used methods to construct such intervals include empirical Bernstein bounds (Maurer and Pontil, 2009), the Hoeffding inequality (Boucheron et al., 2013) for continuous-valued simplex mapping, and Clopper-Pearson methods for discrete mapping (hardmax). These approaches ensure the certified radius is computed with a controlled risk, providing reliable estimates for robustness certification.

Confidence validation for $\mathrm{R}_{\mathrm{mono}}$. In the context of certification using $\mathrm{R}_{\mathrm{mono}}$, the radius is determined based on the index i_1, which is obtained from independent samples (Cohen et al., 2019). The certified radius is defined as:

$$\mu = \mathrm{R}_{\mathrm{mono}}(\underline{\hat{\mathbf{p}}_{i_1}}) \ ,$$

where $\underline{\hat{\mathbf{p}}_{i_1}}$ represents the lower bound of the $(1-\alpha)$-exact Clopper-Pearson confidence interval for the observed proportion $\hat{\mathbf{p}}_{i_1}$. The confidence test for this certification involves the following, H_0: the probability of the certified radius being at least μ is at least $1-\alpha$, i.e.,

$$H_0 : \mathbb{P}(\mathrm{R}_{\mathrm{mono}}(\mathbf{p}_{i_1}) \geq \mu) \geq 1 - \alpha \ .$$

H_1: the probability of the certified radius being less than μ is less than α, i.e.,

$$H_1 : \mathbb{P}(\mathrm{R}_{\mathrm{mono}}(\mathbf{p}_{i_1}) < \mu) < \alpha \ .$$

This framework helps to determine whether the certified radius provides a reliable measure of robustness under the specified risk level α.

For the mono-class certification, the confidence interval is constructed for a single class, making it straightforward to manage the risk level. However, constructing confidence intervals for $\mathrm{R}_{\mathrm{mult}}$ requires extra caution, as it involves managing several interdependent quantities. Previous approaches (Scholten et al., 2023; Voráček and Hein, 2023; Weber et al., 2023; Voráček, 2024) introduce errors in how they allocate the risk budget α. This flawed allocation breaks the Bonferroni correction by assuming dependence between variables, as illustrated by a counter example discussed in the Appendix C.1.

Confidence validation for $\mathrm{R}_{\mathrm{mult}}$. When certifying $\mathrm{R}_{\mathrm{mult}}$, we need to control the overall error rate across multiple comparisons, as we are dealing with several confidence intervals for different classes. The Bonferroni correction is used here to adjust the significance level, ensuring valid hypothesis testing even with dependent tests (Dunn, 1961; Hochberg and Tamhane, 1987; Benjamini and Hochberg, 1995).

Theorem 6.1.2 (Hochberg and Tamhane, 1987)**.**
*Let $H_1, H_2, \ldots, H_m$ be a family of m null hypotheses with p-values $\mathbf{p}_i$, and let α be the desired family-wise error rate. The family-wise error rate (*FWER*) is defined as*

$$\mathrm{FWER} = \mathbb{P}\left(\bigcup_{i=1}^{m} \{\textit{reject } H_i \mid H_i \textit{ is true}\}\right),$$

representing the probability of making at least one Type I error among the multiple tests. The Bonferroni correction sets the individual significance level for each test to $\frac{\alpha}{m}$, such that.

$$(H_i) \textit{ is rejected if } \mathbf{p}_i \leq \frac{\alpha}{m} \ .$$

This ensures control of the family-wise error rate at level α, even for dependent tests.

To handle the multiple comparisons in our problem, we choose $\alpha' = \frac{\alpha}{c}$, where c is the number of classes, thereby adjusting each risk level to control the overall error rate. This adjustment allows us to validate $(1-\alpha)$-level confidence intervals for all classes simultaneously. We define

$$\mu = \mathrm{R}_{\mathrm{mult}}(\underline{\hat{\mathbf{p}}_{I_1}}, \overline{\hat{\mathbf{p}}_{I_2}}) = \frac{\sigma}{2}\left(\Phi^{-1}(\underline{\hat{\mathbf{p}}_{I_1}}) - \Phi^{-1}(\overline{\hat{\mathbf{p}}_{I_2}})\right),$$

where I_1 is the index of the highest lower bound $\underline{\hat{\mathbf{p}}_i}$ among the $(1-\alpha')$-level Clopper-Pearson confidence intervals, and I_2 is the index of the second-highest upper bound $\overline{\hat{\mathbf{p}}_i}$, with $i \neq I_1$.

By constructing these intervals at level $1-\alpha'$, Theorem 6.1.2 ensures that the probability of any interval failing to cover the true parameter $\mathbf{p}_i$ is at most α, thus validating our confidence validations. Similarly:

$$H_0 : \mathbb{P}\left(\mathrm{R}_{\mathrm{mult}}(\mathbf{p}) \geq \mu\right) \geq 1-\alpha$$
$$H_1 : \mathbb{P}\left(\mathrm{R}_{\mathrm{mult}}(\mathbf{p}) < \mu\right) < \alpha \ .$$

The choice of $\alpha' = \frac{\alpha}{c}$ ensures correct coverage across multiple intervals. Using a different value, such as the often picked $\alpha' = \frac{\alpha}{2}$ (Scholten et al., 2023; Voráček and Hein, 2023; Weber et al., 2023; Voráček, 2024), would lead to incorrect coverage assumptions, as shown by a counterexample in the Appendix involving a simple multinomial distribution for $\mathbf{p}$, where dependence assumptions among the confidence intervals are invalid.

6.1.3 Reducing computational overhead with class partitioning method (CPM)

Despite advances in robustness certification, a significant gap persists between theoretical bounds and empirical robustness. This section introduces novel methods to reduce this gap, including an efficient procedure for constructing tighter confidence intervals using class partitioning and a new certificate based on Lipschitz continuity. The robustness certification measure $\mathrm{R}_{\mathrm{mult}}$ performs better for smaller values of $\hat{\mathbf{p}}_{i_1}$,

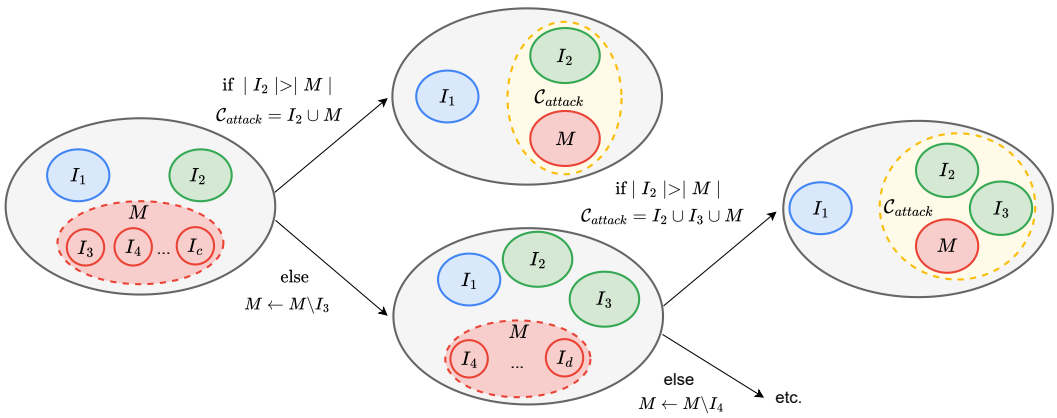


Fig. 6.3.: Illustration of the initial phase of the class partitioning method (CPM). The number of counts in class I is noted $|I|$.

but it requires careful handling of confidence intervals and a conservative risk level $\alpha' = \frac{\alpha}{c}$, where c is the number of classes. This conservative choice can be a limitation

in cases with a large number of classes, such as ImageNet ($c = 1000$), and a limited number of samples.

Algorithm 8 Class partitioning method (CPM) for optimized Bonferroni correction

Require: Initial sample size n_0, estimation sample size n, risk level α

Initialize: Class set $\mathcal{C} = \{1, 2, \ldots, c\}$

Initial Sampling Phase:

Draw n_0 samples and compute the selection counts $C_{\text{select}}[i]$ for each class $i \in \mathcal{C}$

$I_1 \leftarrow \arg\max_{i \in \mathcal{C}} C_{\text{select}}[i]$ ▷ Index of the class with the highest count

$I_2 \leftarrow \arg\max_{i \in \mathcal{C} \setminus \{I_1\}} C_{\text{select}}[i]$ ▷ Index of the class with the second-highest count

Initialize meta-class $M \leftarrow \mathcal{C} \setminus \{I_1, I_2\}$

Define $\mathcal{C}_{\text{attack}} \leftarrow \{I_2, M\}$

while $\sum_{i \in M} C_{\text{select}}[i] > C_{\text{select}}[I_2]$

    $k \leftarrow \arg\max_{i \in M} C_{\text{select}}[i]$

    Remove class k from M

    Add class k to $\mathcal{C}_{\text{attack}}$

Let $\mathcal{P}$ be the partitioning to form $\mathcal{C}_{\text{attack}}$

Estimation Phase:

Draw n samples and compute the selection counts $C_{\text{estim}}[i]$ for each class $i \in \mathcal{C}$

Apply partitioning $\mathcal{P}$ to C_{estim} to obtain bucket counts

Compute $\hat{\mathbf{p}}_{I_1} = \dfrac{C_{\text{estim}}[I_1]}{n}$

for each bucket B in $\mathcal{C}_{\text{attack}}$

    if B is a single class k

        Compute $\hat{\mathbf{p}}_k = \dfrac{C_{\text{estim}}[k]}{n}$

    else ▷ B is meta-class M

        Compute $\hat{\mathbf{p}}_M = \dfrac{\sum_{i \in M} C_{\text{estim}}[i]}{n}$

Compute the total number of buckets $c^\star = |\mathcal{C}_{\text{attack}}| + 1$

Compute the adjusted significance level $\alpha' = \dfrac{\alpha}{c^\star}$

Compute the lower confidence bound $\underline{\hat{\mathbf{p}}}_{I_1}$ with risk α'

for each bucket B in $\mathcal{C}_{\text{attack}}$

    Compute the upper confidence bound $\overline{\hat{\mathbf{p}}}_B$ with risk α'

Let $\overline{\hat{\mathbf{p}}}_{\max} = \max_{B \in \mathcal{C}_{\text{attack}}} \overline{\hat{\mathbf{p}}}_B$

Output: Robustness radius $R(\underline{\hat{\mathbf{p}}}_{I_1}, \overline{\hat{\mathbf{p}}}_{\max})$

To address the challenge of efficiently certifying robustness in multi-class classification, we propose a Class Partitioning Method (CPM) that integrates a structured partitioning strategy with multiple hypothesis testing using the Bonferroni correction, as detailed in Algorithm 8. This method improves statistical power while reducing the computational cost of evaluating each class individually, making it particularly useful for large-scale settings such as ImageNet ($c = 1000$). The CPM algorithm consists of two main phases: initial sampling and partitioning, followed by confidence-bound estimation.

Initial sampling and class partitioning The algorithm begins by drawing an initial sample of size n_0 and computing the selection counts $C_{\text{select}}[i]$ for each class i in the set $\mathcal{C} = \{1, 2, \ldots, c\}$. The two classes with the highest selection counts are identified as I_1 the most probable class and I_2 the second most probable class. The remaining classes are grouped into a meta-class $M = \mathcal{C} \setminus \{I_1, I_2\}$. This setup reduces the number of direct comparisons by treating lower-probability classes as a single entity. The set of classes to attack is initialized as: $\mathcal{C}_{\text{attack}} = \{I_2, M\}$. To refine the partitioning, whenever the total selection count of classes in M exceeds that of I_2, the algorithm iteratively:

(i) removes from M the class k with the highest selection count, and

(ii) adds it to $\mathcal{C}_{\text{attack}}$.

This procedure repeats until the counts are balanced, yielding a partition $\mathcal{P}$ that reduces the number of comparisons while preserving statistical power.

This strategy is visually illustrated in Figure 6.3, where we show how the iterative reallocation of classes improves efficiency.

Estimation phase and confidence bounds After partitioning, the algorithm draws an additional sample of size n and computes the selection counts $C_{\text{estim}}[i]$ for each class i. Using the partitioning $\mathcal{P}$, these counts are grouped into buckets, with empirical probabilities computed as:

$$\hat{\mathbf{p}}_{I_1} = \frac{C_{\text{estim}}[I_1]}{n}, \quad \hat{\mathbf{p}}_B = \frac{C_{\text{estim}}[B]}{n}, \quad \forall B \in \mathcal{C}_{\text{attack}}.$$

The total number of buckets is defined as: $c^\star = |\mathcal{C}_{\text{attack}}| + 1$. To control the family-wise error rate, we apply the Bonferroni correction at the bucket level, adjusting the risk: $\alpha' = \frac{\alpha}{c^\star}$. This reduces conservativeness compared to the naive approach of $\alpha' = \frac{\alpha}{c}$, resulting in tighter confidence intervals without sacrificing statistical guarantees. Using the adjusted risk level, the algorithm computes the lower confidence bound $\underline{\hat{\mathbf{p}}}_{I_1}$ for I_1 and the upper confidence bounds $\overline{\hat{\mathbf{p}}}_B$ for each bucket $B \in \mathcal{C}_{\text{attack}}$. The maximum upper bound across all attackable classes is then determined as: $\overline{\hat{\mathbf{p}}}_{\max} = \max_{B \in \mathcal{C}_{\text{attack}}} \overline{\hat{\mathbf{p}}}_B$. Finally, the algorithm computes and outputs the robustness radius: $\mathrm{R}(\underline{\hat{\mathbf{p}}}_{I_1}, \overline{\hat{\mathbf{p}}}_{\max})$, which quantifies the classifier's robustness against adversarial attacks.

Empirically, this partitioning approach significantly reduces the number of comparisons while improving the tightness of certified robustness bounds. By dynamically

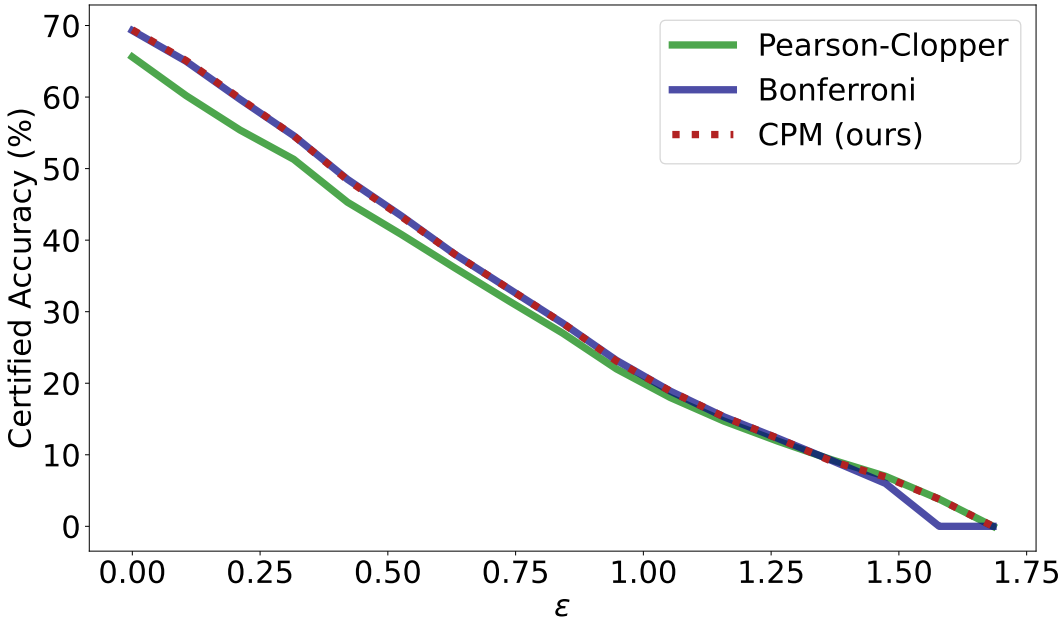


Fig. 6.4.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 0.5$ on the CIFAR-10 dataset, using ResNet-110 trained with noise injection ($\sigma = 0.5$). The plot contrasts our CPM method with Bonferroni and Pearson-Clopper intervals across different perturbation levels ϵ.

transitioning from $\alpha' = \frac{\alpha}{c}$ to $\alpha' = \frac{\alpha}{c^\star}$, the algorithm achieves more efficient robustness guarantees.

6.1.4 Experiment

For all experiments, we use a subset of 500 images for ImageNet. The risk level is set to $\alpha = 0.001$, with $n = 10^4$ samples and $n_0 = 100$ used for class I_1 estimation. All experiments are conducted using a single A100 GPU.

We evaluate our methods on the task of image classification using the CIFAR-10 dataset, employing a ResNet-110 architecture trained with noise injection ($\sigma = 0.5$). The randomized smoothing procedure is applied by adding Gaussian noise to the inputs, providing probabilistic robustness guarantees. We report certified accuracy as a function of the level of ℓ_2 perturbation ϵ, which quantifies the size of adversarial perturbations against which the model is certified to be robust. For certification, we use the certified radius $\mathrm{R}_{\mathrm{mult}}$ for CPM and the standard Bonferroni correction over c classes, while $\mathrm{R}_{\mathrm{mono}}$ is used for Pearson-Clopper intervals. Our CPM method typically produces an effective number of classes $c^\star$ between 2 and 3 providing a less conservative approach to certification. These gains are registered with no additional computational cost compared to the inference cost of the network. As shown in Figure 6.4, our CPM method achieves a balance between the strong performance

of Bonferroni intervals at high perturbation levels and the robustness of Pearson-Clopper intervals at lower perturbations. This demonstrates its effectiveness in providing reliable certified accuracy across a wide range of ϵ values.

To demonstrate the scalability of our method to a larger number of classes, we evaluate it on the ImageNet-1K dataset (1000 classes) using ResNet-50. As shown in Figure 6.5, our CPM method provides tighter confidence intervals compared to the Bonferroni and Pearson-Clopper methods. Typically, CPM yields an effective number of classes, $c^\star$, between 3 and 5. This improvement is particularly pronounced at both low and high perturbation levels, where CPM achieves a more accurate estimate of the certified accuracy. These results highlight the effectiveness of our approach in providing reliable robustness guarantees for large-scale classification tasks.

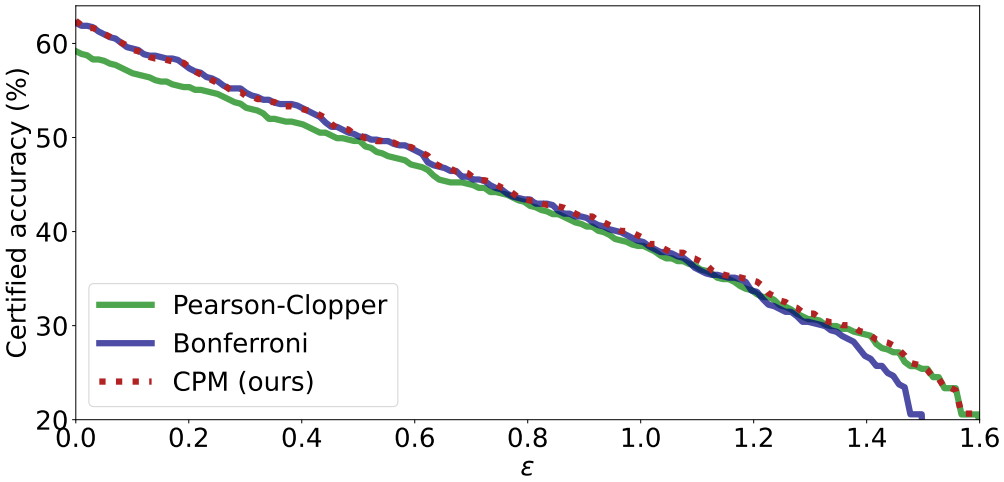


Fig. 6.5.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 0.5$ on the ImageNet dataset, using ResNet-50 trained with noise injection ($\sigma = 0.5$). The plot contrasts our CPM method with Bonferroni and Pearson-Clopper intervals across different perturbation levels ϵ.

We provide additional results for different smoothing standard deviations in the Appendix C.2. For high smoothing standard deviation $\sigma = 1$ on CIFAR-10 CPM provides similar results to Pearson-Clopper while for ImageNet the gain remains significant.

We demonstrate the effectiveness of our CPM method in providing tighter confidence intervals for certification by reducing the number of classes in the design of the certified radius, however, the margin and the variance are not taken into account in this experiment. We will further investigate the impact of these factors in the next section.

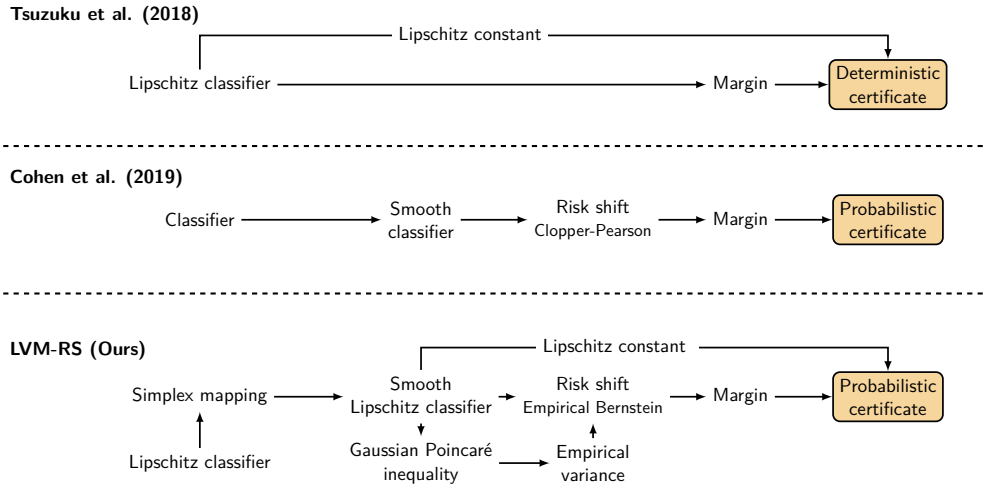


Fig. 6.6.: **First**, Tsuzuku et al., 2018 proposes a deterministic certificate starting from a Lipschitz *base subclassifier*, followed by margin calculation and radius binding. **Second**, Cohen et al., 2019 introduces a *base subclassifier* to create a *smoothed subclassifier*. The risk factor α is then estimated using the Clopper-Pearson interval to provide a probabilistic certificate. **Third**, our method (the *Lipschitz-Variance-Margin Randomized Smoothing* or LVM-RS) extends a smoothed classifier constructed with a Lipschitz base classifier composed with a map which transforms logit to probability vector in simplex. The regularization of the Lipschitz constant is motivated by the Gaussian-Poincaré inequality in Theorem 6.2.1. The empirical variance is applied to the Empirical Bernstein inequality in Proposition 6.2.3 to accommodate for the risk factor α, in the same flavor as in Levine et al., 2019. The pipeline also ends with a probabilistic certificate, similar to the methodology used in Cohen et al., 2019's certified approach.

6.2 Lipschitz-variance-margin tradeoff

We introduce the *Lipschitz-Variance-Margin Randomized Smoothing* (LVM-RS) procedure, illustrated in Figure 6.6. This method is built upon the observation of a fundamental tradeoff between three core quantities underlying robustness certification: the Lipschitz constant of the network, the variance of Monte Carlo (MC) estimates, and the decision margin. This tradeoff arises both in deterministic settings—such as Lipschitz-constrained networks—and in stochastic methods like randomized smoothing, where the classifier is a smoothed version of a base Lipschitz function. All such models share an intrinsic Lipschitz structure, but recognizing and formalizing this tradeoff reveals the underlying quantities that govern certified robustness. We use this theoretical lens to design LVM-RS, a method that tackles the variance component directly. Specifically, we show that controlling the Lipschitz constant reduces the variance of the MC estimate, and we leverage Bernstein's inequality to obtain tighter high-confidence bounds on the risk α. Additionally, we introduce a temperature-controlled simplex mapping to balance margin preservation and variance reduction. This regularized mapping mitigates the tradeoff, ensuring that robustness guarantees remain meaningful even under high noise levels.

6.2.1 Lipschitz control for variance reduction

Lipschitz continuity plays an important role in the sampling process, which is crucial for accurately estimating the smoothed classifier $\tilde{F}$. Specifically, by minimizing the local Lipschitz constant of a classifier F, one can reduce its variance. The following theorem illustrates this relationship for any σ.

Theorem 6.2.1 (Gaussian Poincaré inequality (Boucheron et al., 2013))**.**
Let $Z = (Z_1, \ldots, Z_n)$ represent a vector of i.i.d Gaussian random variables with variance σ^2. For any continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, the variance is given by:

$$\mathbb{V}[h(Z)] \leq \sigma^2 \, \mathbb{E}\left[\|\nabla h(Z)\|^2\right] .$$

We use the latter theorem to immediately derive:

Corollary 6.2.2.
With same hypothesis as Theorem 6.2.1, if h exhibits Lipschitz continuity, we have that:

$$\mathbb{V}[h(Z)] \leq \sigma^2 \, \mathrm{L}(h)^2 .$$

Applying the above corollary to the classifiers $\{F_k\}$ which can be considered differentiable almost everywhere, it is evident that constraining the Lipschitz constant, $L(F_k)$, leads to a diminished variance for F_k. This, in turn, results in a more precise estimation of $\mathbb{E}[F(\mathbf{x} + \delta)]$, as captured by $\frac{1}{n}\sum_{i=1}^{n} F(\mathbf{x} + \delta_i)$. Lowering the local Lipschitz constraints can significantly attenuate the variance and improve the certification results, but it can be too restrictive and cause a drop in performance. To enforce low Lipschitz and reduce performance loss, Cohen et al., 2019 proposed the injection of Gaussian noise during training, Salman et al., 2019 introduced SmoothAdv, which involves adversarial training of the smoothed classifier $\tilde{F}$, to reduce its local Lipschitz constant. The work of Pal and Sulam, 2023 studied how the noisy training on the *sub-classifier* affects the performance and robustness of the *smoothed classifier*. Other noteworthy methods include those by Salman et al., 2020; Carlini et al., 2023, which combine a conventional classifier with a denoiser diffusion model, ensuring that the resulting architecture remains invariant to Gaussian noise, thereby giving Lipschitz continuity to the classifier and preserved performance.

6.2.2 Statistical risk management for low variance

To effectively leverage low-variance estimators in randomized smoothing, one must rely on confidence intervals or concentration inequalities where the variance plays a central role in risk control.

The Clopper–Pearson interval, tailored for binomial proportions, provides an exact α-level confidence interval and is commonly used to compute lower and upper bounds $\underline{\hat{\mathbf{p}}}$ and $\overline{\hat{\mathbf{p}}}$, as in Cohen et al. (2019) and Carlini et al. (2023). This interval naturally applies when the randomized classifier outputs are obtained via a *hardmax* mapping, resulting in Bernoulli trials in Monte Carlo sampling. By contrast, methods such as Lecuyer et al., 2019 and Levine et al., 2019, which smooth scalar outputs in $[0, 1]$ rather than discrete class labels, do not generate Bernoulli outcomes and thus cannot use the Clopper–Pearson interval. Instead, they rely on *Hoeffding's inequality*, which also guarantees an α-level coverage but does not exploit the variance of the estimate. Other inequalities such as the *sub-Gaussian* bound, related to the Gaussian–Poincaré inequality (Massart, 2007), incorporate the Lipschitz constant $\mathrm{L}(F)$ of the classifier. However, this approach has practical limitations: computing $\mathrm{L}(F)$ is NP-hard in general for neural networks, and worst-case Lipschitz bounds often drastically overestimate the true empirical variance. To overcome these issues, we advocate for the use of the *Empirical Bernstein inequality*, particularly when the empirical variance is low. This inequality yields tighter confidence intervals by explicitly incorporating observed variance, and still provides high-probability guarantees for the risk level α. While mentioned in Lecuyer et al., 2019, we build on this idea and integrate it directly into our certification framework.

Proposition 6.2.3 (Empirical Bernstein's inequality (Maurer and Pontil, 2009))**.** *Let $Z_0, Z_1, \ldots, Z_n$ be i.i.d random variables with values between 0 and 1. The risk level is denoted as $\alpha \in [0, 1]$. Then with probability at least $1 - \alpha$ in vector $Z = (Z_1, \ldots, Z_n)$, we have*

$$\mathbb{E}[Z_0] - \frac{1}{n}\sum_{i=1}^{n} Z_i \leq \sqrt{\frac{2S_n(Z)\log(2/\alpha)}{n}} + \frac{7\log(2/\alpha)}{3(n-1)}\,. \tag{6.1}$$

Here, $S_n(Z)$ represents the sample variance $\frac{1}{n(n-1)}\sum_{1\leq i<j\leq n}(Z_i - Z_j)^2$. Note that the bound is symmetric around $\mathbb{E}Z_0$.

In our setting, we apply the Empirical Bernstein inequality to the random variables $Z_i = F_k(\mathbf{x} + \delta_i)$. This formulation provides the flexibility to smooth a wide range of simplex mappings s, beyond the traditional $\mathrm{hardmax}$, thereby enabling better control of the tradeoff between margin and variance. Crucially, it allows us to construct

exact confidence intervals for scalar outputs $F_k(\mathbf{x}+\delta)$, regardless of whether the distribution is discrete or continuous. Also it allow to leverage the variance of the empirical distribution, which is particularly useful when the variance is low, as it leads to tighter bounds than those provided by Hoeffding's inequality. See Fig. C.4.1, for a comparison between Hoeffding's and Bernstein's inequalities.

6.2.3 Scaled simplex for high margin and low variance

The $\mathrm{hardmax}$ mapping maximizes class margins by assigning all mass to the predicted class, yielding the largest possible margins on the simplex. However, $\mathrm{hardmax}$ is not Lipschitz continuous, and as shown in Example C.3, it can lead to significant variance amplification. In contrast, the $\mathrm{softmax}$ mapping smooths class margins, reducing the difference between logits. While this compresses margins, $\mathrm{softmax}$ maintains 1-Lipschitz continuity, preventing variance explosion. To address the limitations of $\mathrm{softmax}$ and $\mathrm{hardmax}$, Martins and Astudillo, 2016 introduced $\mathrm{sparsemax}$, a novel projection that is 1-Lipschitz and yields margins larger than $\mathrm{softmax}$ but smaller than $\mathrm{hardmax}$. This mapping promotes sparsity in the output distribution compared to $\mathrm{softmax}$, reducing variance through contraction, as formalized in Corollary 6.2.2.

To further refine the variance-margin trade-off, we introduce temperature scaling to simplex mappings. For a temperature $T > 0$, we define the scaled mapping $\tau^T : \mathbb{R}^c \to \Delta^{c-1}$ as:

$$\tau^T(\mathbf{z}) = \tau\left(\frac{\mathbf{z}}{T}\right).$$

By adjusting the temperature T, we interpolate between $\mathrm{softmax}$ and $\mathrm{hardmax}$, or between $\mathrm{sparsemax}$ and $\mathrm{hardmax}$. This temperature scaling allows for the selection of an optimal simplex mapping that balances variance reduction with margin maximization, addressing the variance-margin trade-off. The effect of temperature tuning is illustrated in Figure C.4.2.

6.2.4 LVM-RS inference procedure

Given a trained neural network f, we choose a simplex mapping τ from a set of simplex maps $\mathcal{T}$ and a temperature $T \in [T_{\text{lower}}, T_{\text{upper}}]$, defining an ensemble of smoothed soft classifiers $\{\tilde{F}^{\tau^T}\}$:

$$\tilde{F}^{\tau^T}(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\left[\tau_k^T(f(\mathbf{x}+\boldsymbol{\delta}))\right].$$

We begin by sampling n_0 i.i.d. Gaussian perturbations $\delta_i \sim \mathcal{N}(0, \sigma^2 I)$, and compute the smoothed output $Z_i = F_k(\mathbf{x} + \delta_i) \in [0,1]$ for each sample. We then form the empirical estimate:

$$\hat{\mathbf{p}}_k^{\tau^T} := \frac{1}{n_0} \sum_{i=1}^{n_0} Z_i \quad \text{with variance} \quad \hat{v}_k^{\tau^T} := \frac{1}{n_0(n_0-1)} \sum_{i=1}^{n_0} (Z_i - \hat{\mathbf{p}}_k^{\tau^T})^2.$$

Applying the empirical Bernstein inequality (Equation 6.1) with risk level α, we obtain a high-probability lower bound $\underline{\mathbf{p}}_k^{\tau^T}$ and upper bound $\overline{\mathbf{p}}_k^{\tau^T}$ on the true mean:

$$\underline{\mathbf{p}}_k^{\tau^T} = \hat{\mathbf{p}}_k^{\tau^T} - \sqrt{\frac{2\hat{v}_k^{\tau^T} \log(2/\alpha)}{n_0}} - \frac{7\log(2/\alpha)}{3(n_0-1)},$$

$$\overline{\mathbf{p}}_k^{\tau^T} = \hat{\mathbf{p}}_k^{\tau^T} + \sqrt{\frac{2\hat{v}_k^{\tau^T} \log(2/\alpha)}{n_0}} + \frac{7\log(2/\alpha)}{3(n_0-1)}.$$

We denote by $\underline{\mathbf{p}}_{I_1}^{\tau^T}$ and $\overline{\mathbf{p}}_{I_2}^{\tau^T}$ the corrected probabilities associated respectively with the predicted class I_1 and the highest non-predicted class I_2. These corrected estimates are then used to compute the certified radius $\mathrm{R}_{\mathrm{mult}}(\underline{\mathbf{p}}_{I_1}^{\tau^T}, \overline{\mathbf{p}}_{I_2}^{\tau^T})$, which accounts for estimation uncertainty at level α.

Finally, to maximize robustness, we perform a grid search over temperature and smoothing parameters and select the optimal pair:

$$(\tau^\star, T^\star) = \arg\max_{\tau, T} \mathrm{R}_{\mathrm{mult}}(\underline{\mathbf{p}}_{I_1}^{\tau^T}, \overline{\mathbf{p}}_{I_2}^{\tau^T}).$$

Then, a sampling of size n is performed and we evaluate an MC estimate of $\tilde{F}^{\tau^{\star T^\star}}(\mathbf{x})$, which gives $\hat{\mathbf{p}}^\star$ and associated risk-corrected $\underline{\mathbf{p}_{I_1}}^\star$ and $\overline{\mathbf{p}_{I_2}}^\star$. We return the prediction I_1 and associated certified radius $\mathrm{R}_{\mathrm{mult}}(\underline{\mathbf{p}_{I_1}}^\star, \overline{\mathbf{p}_{I_2}}^\star)$.

6.2.5 Experiment on certified accuracy with LVM-RS

In this experiment, we empirically validate the efficacy of our proposed inference procedure presented in Algorithm 9. Our approach leverages the variance-margin trade-off to achieve state-of-the-art RS results. We use the following parameters: temperatures: $T_{\mathrm{lower}} = 0.01$, $T_{\mathrm{upper}} = 50$ (50 values) and implex maps: $\mathcal{T} = \{\mathrm{sparsemax}, \mathrm{softmax}, \mathrm{hardmax}\}$.

Algorithm 9 LVM-RS $(f, \sigma, \mathbf{x}, n_0, n)$

$\text{samples}_{n_0} \leftarrow \text{sample\_scores}(f, \mathbf{x}, n_0, \sigma)$ ▷ validation set, $n_0 \times c$
$\text{samples}_{n} \leftarrow \text{sample\_scores}(f, \mathbf{x}, n, \sigma)$ ▷ certification set, $n \times c$
for $T \in [T_{\text{lower}}, T_{\text{upper}}]$
  for $\tau \in \mathcal{T}$
    $\underline{\mathbf{p}_{I_1}}^{\tau^T} \leftarrow \frac{1}{n_0} \sum_{i=1}^{n_0} \tau^T(\text{samples}_{n_0}[i,:]) - \text{shift}(S_{n_0}(\tau^T(\text{samples}_{n_0})), \alpha, n_0)$
    $\overline{\mathbf{p}_{I_2}}^{\tau^T} \leftarrow \frac{1}{n_0} \sum_{i=1}^{n_0} \tau^T(\text{samples}_{n_0}[i,:]) + \text{shift}(S_{n_0}(\tau^T(\text{samples}_{n_0})), \alpha, n_0)$
$(\tau^\star, T^\star) \leftarrow \arg\max_{\tau,T} \text{R}_{\text{mult}}\left(\underline{\mathbf{p}_{I_1}}^{\tau^T}, \overline{\mathbf{p}_{I_2}}^{\tau^T}\right)$
$\underline{\mathbf{p}_{I_1}}^{\star} \leftarrow \frac{1}{n_0} \sum_{i=1}^{n_0} \tau^{\star T^\star}(\text{samples}_{n}[i,:]) + \text{shift}(S_{n}(\tau_\star^{t^\star}(\text{samples}_{n})), \alpha, n)$ ▷ dim. c
$\overline{\mathbf{p}_{I_2}}^{\star} \leftarrow \frac{1}{n_0} \sum_{i=1}^{n_0} \tau^{\star T^\star}(\text{samples}_{n}[i,:]) - \text{shift}(S_{n}(\tau_\star^{t^\star}(\text{samples}_{n})), \alpha, n)$ ▷ dim. c
return prediction $\arg\max_k \bar{\mathbf{p}}_k^\star$ and certified radius $\text{R}_{\text{mult}}(\underline{\mathbf{p}_{I_1}}^{\star}, \overline{\mathbf{p}_{I_2}}^{\star})$

Tab. 6.2.: Best certified accuracies across $\sigma \in \{0.25, 0.5, 1.0\}$ for different levels of perturbation ϵ, on CIFAR-10, for $n = 10^5$ samples and risk $\alpha = 1\text{e}{-3}$.

Methods	Best certified accuracy (ε)					Average time (s)
	0.0	0.25	0.5	0.75	1.0	
Carlini et al.	86.72	74.41	58.25	40.96	29.91	7.10
LVM-RS (ours)	**88.49**	**76.21**	**60.22**	**43.76**	**32.35**	7.11

In this experiment, we empirically validate the efficacy of our proposed inference procedure presented in Algo. 9, highlighting its capability to improve randomized smoothing and achieve certified accuracy. Central to our approach is the leveraging of the variance-margin tradeoff, which as we demonstrate, yields state-of-the-art RS results. We further showcase how the procedure enhances the off-the-shelf state-of-the-art baseline model of Carlini et al., 2023, which utilizes a vision transformer coupled with a denoiser for randomized smoothing. Note that the choice of simplex mapping and temperature depends on the input. The baseline consists of the model of Carlini et al., 2023 which does smoothing of hardmax of *base classifier* and uses the Pearson-Clopper confidence interval to control the risk α.

To compare the baseline with our method, certified accuracies are computed with R_{mult} in the function of the level of perturbations ϵ, for different noise levels $\sigma \in \{0.25, 0.5, 1\}$. Results are presented in Figure C.5.1 for CIFAR-10 and in Figure C.5.2 for ImageNet. We see that our method increases results, especially in the case of high σ, in the case of $\sigma \in \{0.5, 1.0\}$ the overall certified accuracy curve in the function of ϵ the maximum perturbation is lifted towards higher accuracies. Results are presented in Table 6.2 for CIFAR-10 and in Table 6.3 for ImageNet. Computation was performed on GPU V100, reported average time is the computational cost of one input $\mathbf{x}$ proceeds by RS and LVM-RS, we see that the computation gap between the two methods is narrow for CIFAR10 but is a bit wider for ImageNet. Detailed results are presented in Appendix C.5.

Tab. 6.3.: Best certified accuracies across $\sigma \in \{0.25, 0.5, 1.0\}$ for different levels of perturbation ϵ, on ImageNet, for $n = 10^4$ samples and risk $\alpha = 1\text{e}{-3}$.

Methods	Best certified accuracy (ϵ)						Average time (s)
	0.0	0.5	1.0	1.5	2	3	
Carlini et al.	79.88	69.57	51.55	**36.04**	25.53	14.01	6.46
LVM-RS (ours)	**80.66**	**69.84**	**53.85**	**36.04**	**27.43**	**14.31**	7.03

6.3 Conclusion

This chapter addressed risk management for randomized smoothing (RS) to achieve certified adversarial robustness. We demonstrated the superiority of multi-class certified radii R_{multi} over the mono-class approach R_{mono}, particularly in high-variance scenarios where R_{mono} fails to provide meaningful certification. Our empirical analysis highlighted the increasing gap between R_{multi} and R_{mono} as variance grows.

To mitigate the computational overhead associated with multi-class certification, we introduced the Class Partitioning Method (CPM), which refines Bonferroni corrections by partitioning classes into relevant buckets. This method reduced conservativeness and enhanced certified accuracy without increasing computational complexity. Experiments on CIFAR-10 and ImageNet validated CPM's efficiency, showing notable improvements in certification accuracy at different noise levels.

We further addressed the variance-margin tradeoff by introducing the Lipschitz-Variance-Margin (LVM) tradeoff, which operates at the model level to minimize variance during Monte Carlo sampling. By leveraging scalable simplex mappings and temperature scaling, LVM improves the stability of smoothed classifiers. Our combined RS + LVM approach consistently outperformed baseline methods, achieving state-of-the-art results across perturbation budgets.

In the next chapter, we explore the impact of Lipschitz's constant regularization on neural network generalization. Additionally, we extend randomized smoothing principles to the parameter space rather than the input space, aiming to enhance generalization further.

7 Regularization to enhance generalization

Contents

Generalization remains a fundamental challenge in deep learning, particularly as models grow in complexity and are trained on increasingly large and diverse datasets. Regularization techniques aim to improve generalization by constraining the learning process and guiding models toward solutions that perform well on unseen data. In this chapter, we explore two complementary approaches to regularization: Lipschitz regularization, which constrains the overall sensitivity of a network, and flatness-based regularization, which focuses on minimizing sharpness in the loss landscape.

To enforce Lipschitz regularization, we leverage the Gram iteration method developed in Chapter 3. This technique allows us to efficiently regularize the Lipschitz constants of individual layers, including convolutional layers, by estimating and constraining their spectral norms. By reducing the Lipschitz constant, we enhance the network's robustness and generalization performance.

For flatness-based regularization, we present a theoretical framework to quantify the reduction in the Hessian curvature of the loss landscape with Gaussian smoothing using the result on Weierstrass transform from Chapter 4 and we link flat minima to Lipschitz continuity. We introduce a novel regularization method, activation decay, which directly reduces the sharpness of the loss by smoothing the penultimate activations. This approach not only flattens the loss landscape but also improves generalization and robustness, providing a computationally efficient alternative to existing sharpness-aware techniques.

7.1 Lipschitz-based regularization

The Lipschitz constant of a network plays a critical role in generalization bounds (Bartlett et al., 2017), as it quantifies the sensitivity of the model to variations in the input data, thereby linking generalization performance with robustness. To regularize the Lipschitz constant, we apply spectral norm regularization on both the convolutional filters and dense layers, following the approach of Araujo et al. (2020) and Singla and Feizi (2021a), as described in previous Section 4.2. This technique is analogous to weight decay, which regularizes the Frobenius norm of the weights.

The regularization term added to the loss function is:

$$\mathcal{L}_{\text{reg}} = \mu_{\text{reg}} \sum_{l=1}^{L} \left\| \mathbf{W}^{(l)} \right\|_2 . \tag{7.1}$$

Regularizing each layer individually constrains the overall Lipschitz constant of the network via the product's upper bound. The regularized training loss writes as

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{reg}}.$$

Applying direct PUB as a regularization term is not ideal, as the product of Lipschitz bounds can become unstable and grow unbounded. One alternative is to employ the logarithm of the product, as proposed by Araujo et al., 2020. However, empirical results indicate that summing individual bounds during regularization yields the best performance.

To assess the impact of spectral norm regularization on classification accuracy and training time, we conducted experiments on CIFAR-10 using the ResNet-18 architecture (He et al., 2016). The training setup follows the protocol described in Chapter 4, with the following hyperparameters: 200 epochs, batch size 256, SGD with momentum 0.9, an initial learning rate of 0.1, and a cosine annealing schedule. The baseline model is trained without any regularization (*i.e.* $\mu_{\text{reg}} = 0$).

We compare several spectral norm estimation methods from prior work (Ryu et al., 2019; Araujo et al., 2021; Singla and Feizi, 2021a) against our proposed estimator, denoted `norm2_circ` (see Algorithm 5). Our method relies on explicit backward-mode differentiation (Equation 3.6) to improve both computational efficiency and memory usage.

Regularization terms. We apply two types of regularization to the convolutional layers:

- **Spectral norm regularization:** we penalize the deviation of the spectral norm from a target threshold using

$$\mathcal{L}_{\text{reg}} = \mu_{\text{reg}} \sum_{\ell} \|\mathbf{W}^{(\ell)}\|_2^2,$$

  where $\|\mathbf{W}^{(\ell)}\|_2$ denotes the spectral norm of layer ℓ, and μ_{reg} = 1e-1.

- **Weight decay (WD) / Frobenius norm regularization:** we also apply a standard weight decay regularization

$$\mathcal{L}_{\text{reg}} = \mu_{\text{wd}} \sum_{\ell} \|\mathbf{W}^{(\ell)}\|_F^2,$$

  with μ_{wd} = 5e-3.

On the choice of regularization coefficients. The coefficients μ_{reg} and μ_{wd} are not directly comparable and must be chosen separately. This is because the Frobenius norm scales with the square root of the number of weights in a layer, while the spectral norm corresponds to the largest singular value and is typically of smaller magnitude. Using the same coefficient for both would result in significantly unbalanced regularization strengths.

Additional details. We do not report results from Sedghi et al. (2019), as their exact spectral norm computation is computationally prohibitive—requiring over 6750 seconds per epoch. We also evaluate the effect of the Gram iteration count $t \in 3, 4, 5, 6$ at the end of training. Higher values of t reduce the standard deviation of the spectral norm estimate, with all reported training times corresponding to $t = 6$. For statistical robustness, all experiments are repeated 10 times.

Results of experiments are reported in Table 7.1. Most of the bounds for Lipschitz regularization (Ryu et al., 2019; Araujo et al., 2021; Singla and Feizi, 2021a) tend to slightly degrade accuracy in comparison to baseline, contrary to our bound where a small accuracy gain can be observed: our Lipschitz regularization is not a trade-off on accuracy under this training configuration. Furthermore, the low standard deviation of our accuracies suggests that our regularization stabilizes training contrary to all other approaches. The computational cost of our method is the lightest of all Lipschitz regularization approaches.

To demonstrate the scalability of our method, a ResNet18 is trained on the ImageNet-1k dataset, processing 256×256 images. ResNet18 baseline (with WD) is compared with two regularized ResNet18: one trained from scratch on 88 epochs, and one initialized on a pre-trained baseline and fined-tuned on 10 epochs. Trainings are

Tab. 7.1.: This table shows test accuracies and training times for ResNet18 on CIFAR-10, repeated 10 times. Our method has a much narrower standard deviation, slightly better accuracy, and lower training time in comparison to other bounds for Lipschitz regularization. Here WD denotes weight decay.

	Test accuracy (%)	**Time per epoch (s)**
Baseline	93.32 ± 0.12	11.4
Baseline WD	93.30 ± 0.19	11.4
Ryu et al.	93.27 ± 0.13	95.0
Sedghi et al.	✗	6750
Araujo et al.	90.66 ± 0.26	37.4
Singla and Feizi	93.29 ± 0.15	38.4
Ours	93.48 ± 0.08	31.9

repeated four times on 4 GPU V100. The results of experiments are reported in Table 7.2. This experiment shows that our bound remains efficient when dealing with large images. Moreover, it is not required to train a new ResNet from scratch: any trained ResNet can be fine-tuned on a few epochs to be regularized.

Tab. 7.2.: This table shows test accuracies and training times for ResNet18 on ImageNet-1k, repeated 4 times.

	Test accuracy (%)	**Time per epoch (s)**
Baseline WD	69.76	746
Ours	70.77 ± 0.12	782
Ours (fine-tuned)	70.50 ± 0.05	782

This experiment demonstrates that our method is efficient for large images and can be applied to pre-trained models, allowing for quick fine-tuning with regularization.

One limitation is that our method does not encompass the normalization layers, such as batch normalization (Ioffe and Szegedy, 2015) or layer normalization (Ba et al., 2016), which are often used in deep networks. Penalizing the spectral norm of these layers is not straightforward, as they do not have a well-defined weight matrix, but are non linear transformations of the input.

In the following section, we explore a different approach to regularization, focusing on minimizing sharpness in the loss landscape to improve generalization. This approach is complementary to Lipschitz regularization and can be applied alongside it.

7.2 Flatness regularization through activation decay

The nature of the minima in the loss landscape has often been cited as a key factor influencing generalization. Models that converge to sharp minima tend to be highly sensitive to small perturbations, resulting in poor performance on unseen data. In contrast, models that converge to flatter minima are often associated with improved robustness and generalization (Hochreiter and Schmidhuber, 1997a). Flatness intuitively corresponds to solutions that are less sensitive to small changes in inputs or parameters, making the model more robust to variations. While sharpness-based measures have been shown to correlate strongly with generalization under certain settings (Jiang* et al., 2020), flatness is not a universal guarantee of better generalization. This is because sharpness measures can be sensitive to parameterization and re-scaling effects in deep, overparameterized networks (Zhang et al., 2017). Despite these nuances, regularization techniques aimed at minimizing sharpness or curvature remain effective heuristics for guiding models toward solutions that generalize well in practice.

A widely used regularization method for encouraging flatness is weight decay, or ℓ_2 regularization on weights. This technique penalizes the magnitude of the model's weights by adding a term proportional to the squared norm of the parameters to the loss function. Weight decay serves to constrain large parameter values, effectively smoothing the loss landscape and reducing the complexity of the learned model. By controlling the magnitude of the weights, weight decay helps the model find flatter minima, which is associated with improved generalization (Krogh and Hertz, 1992). In addition to being computationally efficient, weight decay aligns with other regularization strategies, such as Lipschitz constraints, as explored in Section 7.1.

7.2.1 Link between Lipschitz networks and flat minima

We recall the definition of a feed-forward neural network f as the composition of its L layers: $f = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$. Given an input $\mathbf{x} \in \mathbb{R}^d$, we define the activations at each layer as: $\mathbf{h}^{(l)} = f^{(l)}(\mathbf{h}^{(l-1)})$, for $l = 1, \ldots, L$, where $\mathbf{h}^{(0)} = \mathbf{x}$. Each layer applies a linear transformation with weights $\mathbf{W}^{(l)}$, followed by a nonlinearity $\rho^{(l)}$, typically ReLU or GELU:

$$f^{(l)}(\mathbf{h}^{(l-1)}) = \rho^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}\right), \quad l = 1, \ldots, L-1.$$

The final layer applies no nonlinearity ($\rho^{(L)}$ is the identity), producing: $f^{(L)}(\mathbf{h}^{(L-1)}) = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)}$. For a given label $\mathbf{y}$, the loss is denoted by $\mathcal{L}(\mathbf{h}^{(L)}, \mathbf{y}, \theta)$, where $\theta = \text{vec}\left(\{\mathbf{W}^{(l)}\}_{l=1,\dots,L}\right)$.

Penalizing the parameter norm $\|\boldsymbol{\theta}\|_2$ is a common regularization strategy that constrains the model's Lipschitz constant by reducing the product of the spectral norms of the layers. Specifically, the product upper bound of the network's Lipschitz constant can be expressed as:

$$\text{PUB}(f) = \prod_{l=1}^{L} \left\|\mathbf{W}^{(l)}\right\|_2 \leq \prod_{l=1}^{L} \left\|\mathbf{W}^{(l)}\right\|_{\text{F}},$$

where $\|\mathbf{W}^{(l)}\|_2$ and $\|\mathbf{W}^{(l)}\|_F$ denote the spectral and Frobenius norms of the weights in layer l, respectively.

Both sharpness regularization and Lipschitz networks improve generalization by reducing the model's sensitivity to perturbations. Sharpness regularization operates on the loss landscape to promote flatter minima, while Lipschitz regularization constrains the function's overall behavior. Notably, Lipschitz constraints also encourage a regular loss landscape to some extent.

Sharp minima are usually characterised by a large spectral norm of the loss Hessian $\|\nabla^2_\theta \mathcal{L}(\theta)\|_2$. To make curvature control more tractable, we first analyze the structure of the Hessian. The following theorem provides a decomposition of the full Hessian norm $\|\nabla^2_\theta \mathcal{L}(\theta)\|_2$ into a product involving the sensitivity of the penultimate activations with respect to the parameters and the curvature of the loss with respect to these activations.

Theorem 7.2.1 (Layerwise Hessian Norm Decomposition)**.** *Consider the empirical loss $\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{a}_i^{(L)}(\theta), y_i)$, with $\ell(\cdot, y)$ twice differentiable in the logits and all activations 1-Lipschitz. Suppose a minimizer $\theta^\star$ satisfies* logit stationarity, *i.e. $\nabla_{\mathbf{a}^{(L)}} \ell(\mathbf{a}_i^{(L)}(\theta^\star), y_i) = 0$ for all i. Then the spectral norm of the Hessian at $\theta^\star$ satisfies*

$$\|\nabla^2_\theta \mathcal{L}(\theta^\star)\|_2 \leq \underbrace{\left(\sum_{j=1}^{L-1} \left\|\frac{\partial \mathbf{h}^{(j)}}{\partial \theta}(\theta^\star)\right\|_2 \prod_{l=j+1}^{L-1} \|\mathbf{W}^{(l)}\|_2\right)^2}_{\text{Sensitivity of } \mathbf{h}^{(L-1)} \text{ to parameters}} \underbrace{\|\nabla^2_{\mathbf{h}^{(L-1)}} \mathcal{L}(\theta^\star)\|_2}_{\text{Curvature w.r.t. activation } \mathbf{h}^{(L-1)}} \tag{7.2}$$

Find proof in Appendix D.2.2. Given that over-parameterized networks typically reach near-zero training loss (Zhang et al., 2017), assuming logit stationarity at

a minimizer $\theta^\star$ is reasonable: it holds in regression (MSE), in classification with label smoothing, and asymptotically in cross-entropy with one-hot labels as logits diverge. This theorem emphasizes the outsized influence of deeper layers' spectral norms on curvature, highlighting their crucial role in shaping the loss landscape. Regularizing deeper layers is especially effective for controlling sharpness. Weight decay or spectral norm regularization naturally accomplishes this by bounding the spectral norm through the Frobenius norm constraint.

This inequality highlights that the contributions from deeper layers to the Hessian norm bound are more significant, as the spectral norm $\|\mathbf{W}^{(l)}\|_2$ is accumulated more times for deeper layers. Consequently, the curvature of the loss landscape is strongly influenced by the Lipschitz constants of the deeper layers.

The term $\left\|\nabla^2_{\mathbf{h}^{(L-1)}}\mathcal{L}(\boldsymbol{\theta})\right\|_2$ reflects the influence of the Hessian with respect to the penultimate activations. Combined with the accumulated Lipschitz constants, this term governs the overall smoothness of the loss landscape. Managing the spectral norms of intermediate layers $\|\mathbf{W}^{(l)}\|_2$ and the penultimate-layer Hessian offers a more tractable approach than directly constraining the Hessian of the loss function, which would require costly second-order computations. Specifically, for a model with d parameters, the Hessian matrix has size $d \times d$, requiring $O(d^2)$ storage and $O(d^3)$ computation. This makes direct computation infeasible for large networks Goodfellow et al., 2016.

The above theorem demonstrates that Lipschitz networks encourage flatter loss landscapes, which is beneficial for generalization. Weight decay, by bounding the Frobenius norm $\|\mathbf{W}^{(l)}\|_F$, indirectly controls the spectral norm $\|\mathbf{W}^{(l)}\|_2$, providing a practical and computationally efficient regularization strategy. Section 7.1 further explores spectral norm regularization as an efficient method for controlling curvature through Lipschitz regularization. However, weight decay does not control the term $\left\|\nabla^2_{\mathbf{h}^{(L-1)}}\mathcal{L}(\boldsymbol{\theta})\right\|_2$, which motivates the need for additional techniques based on loss smoothing to reduce sharpness in the loss landscape, as discussed in the next section.

7.2.2 Loss smoothing for flat minima

Bishop (1995) first established the connection between noise injection on inputs and deterministic regularization on parameters, demonstrating how noise injection can be cast as a form of regularization. In more recent work, Wei et al. (2020) and Orvieto et al. (2022) further explored this connection by analyzing how specific forms of noise injection, such as anticorrelated noise and dropout (Srivastava et al., 2014), can encourage flatter minima and improve generalization. While noise

injection introduces stochasticity during training, making models more robust to perturbations and guiding them toward flatter minima, those methods introduce randomness into the optimization process, which can lead to performance variability across different runs.

Another important method in the field of regularization is Sharpness-Aware Minimization (SAM) (Foret et al., 2021), see Equation 2.15, which directly targets sharpness by introducing a min-max optimization framework. SAM aims to minimize the worst-case sharpness in a neighborhood of the model's parameters, encouraging convergence to flatter minima. While effective, SAM's computational cost and memory overhead are significantly higher than simpler methods like weight decay and noise injection. The perturbed loss evaluation requires multiple forward-backward. Zhang et al. (2018) show that batch normalization (Ioffe and Szegedy, 2015) and residual connections (He et al., 2016) enhance backpropagation by improving the local Hessian's spectrum leading to better gradient flow. Beyond simply smoothing the loss landscape, methods like SAM provide robustness to label noise by guiding models to converge to flatter minima, where the sensitivity to noisy labels is reduced (Baek et al., 2024).

Inspired by the works of Bishop (1995) and Orvieto et al. (2022), we propose a novel deterministic noise-based regularization that operates on activations rather than weights, with the same low computational cost. This approach reduces sharpness in the loss landscape while maintaining computational efficiency. In the regime near a minimum, where the loss gradient's norm $\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\|_2$ is small, understanding the impact of smoothing techniques like Gaussian noise on the curvature of the loss is crucial for generalization. The following corollary quantifies the reduction in the spectral norm of the Hessian—an indicator of sharpness—when Gaussian noise is applied to the parameters near a local minimum.

Corollary 7.2.2 (Dimension-free bound on Hessian norm of Gaussian smoothed loss)**.**
Let $\mathcal{L}:\mathbb{R}^d\to\mathbb{R}$ be twice differentiable. Suppose that the gradient $\nabla_\theta\mathcal{L}$ is H-Lipschitz continuous, and that for some $\epsilon>0$, $\|\nabla_\theta\mathcal{L}(\theta)\|_2\le\epsilon$. For $\Delta\sim\mathcal{N}(0,\sigma^2 I)$, the spectral norm of the Hessian of the Gaussian smoothed loss is bounded by:

$$\|\nabla_\theta^2\,\mathcal{L}^\sigma(\theta)\|_2 \le \underbrace{H}_{\text{base curvature}} \underbrace{\operatorname{erf}\left(\frac{\epsilon}{\sqrt{2}H\sigma}\right)}_{\text{smoothing term}<1}. \tag{7.3}$$

The proof (see Appendix D.2.1) of this corollary adapts Theorem 4.6.1 which quantifies the regularity of Weierstrass transform when underlying transformed function is

already regular, previously applied in randomized smoothing with Gaussian noise applied to the inputs to obtain Lipschitz continuity. Here Gaussian noise is applied to the network's parameters $\boldsymbol{\theta}$ and it translates to increased flatness of the loss landscape. The work of Nesterov and Spokoiny (2017) was the first to derive bounds on the regularity of the Gaussian smoothed loss. Unlike earlier bounds, which depend on the dimensionality d, this result is dimension-free, making it more suitable for high-dimensional deep networks. The result provides a quantitative measure of how Gaussian noise injection smooths the loss landscape, reducing sharpness and promoting generalization. Notably, while Orvieto et al. (2022) explored the connection between noise injection and the trace of the Hessian, our bound focuses on the Hessian's spectral norm, offering a complementary perspective on sharpness reduction.

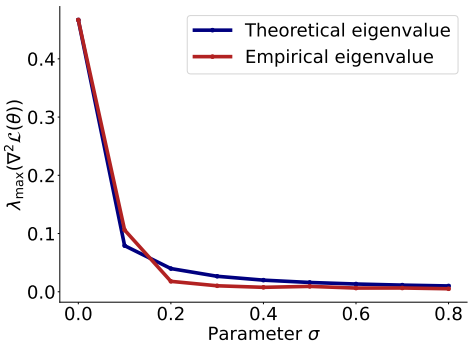


Fig. 7.1.: Comparison between the theoretical bound on the largest eigenvalue of the Hessian given by Corollary 7.2.2 and the empirical value computed with PyHessian with a relative tolerance of $1e{-}3$, on a ResNet-56 model trained on CIFAR-10 for 300 epochs.

Empirical validation of regularization effects This experiment empirically evaluates how closely the theoretical bound from Corollary 7.2.2 aligns with observed Hessian norms. Several runs of training on CIFAR-10 are performed with the ResNet-56 model and different parameters σ for the AD. We use the PyHessian library to compute the Hessian operator norm (Yao et al., 2020). Then, we compute the largest eigenvalue of the Hessian on the final layers of the network to assess the sharpness reduction predicted by Gaussian smoothing. Here Hessian is computed only w.r.t to $\mathbf{W}^{(L)}$ of the final layer. We estimate ϵ by averaging the gradient's norm near the minimum at the end of the training. The Hessian eigenvalue is computed on the training set at the end of the training after 300 epochs.

We see in Figure 7.1, that the theoretical bound gives the correct trend of the evolution of the curvature of the Hessian, the remaining mismatch might come from stochasticity in the Hessian eigenvalue computation, the Jensen gap, or another factor.

7.2.3 From Noise to Regularization: Activation Decay

Building on the curvature decomposition in Theorem 7.2.1 and the smoothing guarantees from Corollary 7.2.2, we now derive a simple and efficient regularization method. The key idea is to contract the activation-level curvature term without injecting noise during training. To this end, we analytically translate the effect of Gaussian smoothing into a deterministic penalty, resulting in a method we call *activation decay*.

Rather than adding noise as in dropout, we consider a Gaussian perturbation $\Delta \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ applied solely to the final-layer weights $\mathbf{W}^{(L)}$. For fixed penultimate activations $\mathbf{h}^{(L-1)}$, this yields a smoothed loss:

$$\mathcal{L}^{\sigma}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)}, \mathbf{y}) = \mathbb{E}_{\Delta}\left[\mathcal{L}\big((\mathbf{W}^{(L)} + \Delta)\mathbf{h}^{(L-1)}, \mathbf{y}\big)\right].$$

This smoothing strategy, closely related to softmax over Gaussian distributions (Lu et al., 2021) and final-layer isolation techniques (Newman et al., 2021), remains fully differentiable and admits a closed-form upper bound for standard losses such as cross-entropy.

Theorem 7.2.3 (Smoothed Cross-Entropy Loss)**.**
Let $\mathcal{L}_{\mathrm{CE}}$ be the cross-entropy loss, $\mathbf{h}^{(L-1)} \in \mathbb{R}^d$ be an input from the penultimate layer, $\mathbf{y} \in \mathbb{R}^c$ a one-hot label vector, and $\mathbf{W}^{(L)} \in \mathbb{R}^{c \times d}$ a weight matrix. Then the following bound holds:

$$\mathcal{L}^{\sigma}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)}, \mathbf{y}) \leq \underbrace{\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)}, \mathbf{y})}_{\textit{base loss}} + \underbrace{\frac{\sigma^2}{2}\|\mathbf{h}^{(L-1)}\|_2^2}_{\textit{smoothing term}}. \tag{7.4}$$

Minimizing this upper bound[1] is equivalent to augmenting the original loss with an ℓ_2 penalty on $\mathbf{h}^{(L-1)}$. We refer to this regularizer as *activation decay* (AD), by analogy with weight decay. Unlike dropout or other stochastic methods, AD is deterministic, low-variance, and introduces no computational overhead during training.

The resulting penalty $\|\mathbf{h}^{(L-1)}\|_2^2$ directly targets the curvature term $\nabla^2_{\mathbf{h}^{(L-1)}}\mathcal{L}$ identified in Theorem 7.2.1, acting as a surrogate for Gaussian smoothing at the classifier level. This construction is rigorously justified by Corollary **??**, which proves that smoothing contracts the Hessian's spectral norm in a dimension-independent way. Meanwhile, the parameter-sensitivity component of curvature can be independently

[1] See Appendix D.1.1 for empirical validation of the tightness of this bound.

controlled via standard techniques such as weight decay or spectral norm regularization. In this view, activation decay and weight regularization act complementarily, each addressing a distinct factor in the Hessian decomposition.

7.2.4 Experiments on vision tasks

We propose to evaluate our method Activation Decay (AD) on several vision tasks.

Classification with MLP on CIFAR-10 This experiment compares different regularization techniques on a 4-layer Multi-Layer Perceptron (MLP) network with GELU activation. The MLP has 3072 input features, and the training was conducted on the CIFAR-10 dataset. The model was trained using Stochastic Gradient Descent (SGD) without momentum, and no weight decay was applied. A learning rate scheduler with annealing was used to adjust the learning rate, which was set to $1\mathrm{e}{-1}$. We use a batch size of 128 and standard data augmentation techniques, including random horizontal flips and random crops with padding of 4 pixels. These augmentations are applied to the training data to improve generalization. Each experiment was repeated 10 times to ensure statistical significance.

We explored various regularization methods: We apply dropout (DO), parameterized by probability p, on intermediate layers $\mathbf{h}^{(l)}$; weight decay (WD) on all layers, parameterized by σ; activation decay (AD), parameterized by σ, on the last layer $\mathbf{h}^{(L)}$; a combination of AD on the last layer $\mathbf{h}^{(L)}$, parameterized by σ, with weight decay on intermediate layers $\mathbf{h}^{(l)}$, where the best parameter is obtained from the previous weight decay experiment, we perform the same process replacing weight decay with spectral norm regularization to showcase its combination with AD; and SAM parametrized by ρ. Results are presented in Figure 7.2. The baseline is at 62.17%.

Our results demonstrate that AD increases generalization when $\sigma = 0.1$, improving accuracy by 2.63 %. Applying dropout alone also slightly enhances generalization, improving by 1.73% for best parameter $p = 0.1$. However, combining dropout with AD does not yield additional benefits and performs worse than using AD alone. The method that combines AD and weight decay performs better than only AD as highlighted by Theorem 7.2.1. , following this result, we try to combine AD and spectral norm regularization of layers as it is a closer upper bound to the formula from Theorem 7.2.1 and it provides the best-obtained result. This result confirms the impact of the Lipschitz-constrained network (through spectral norm regularization of intermediate layers) and its link to overall flatness and impact on generalization. SAM does not provide as good results for the MLP architecture

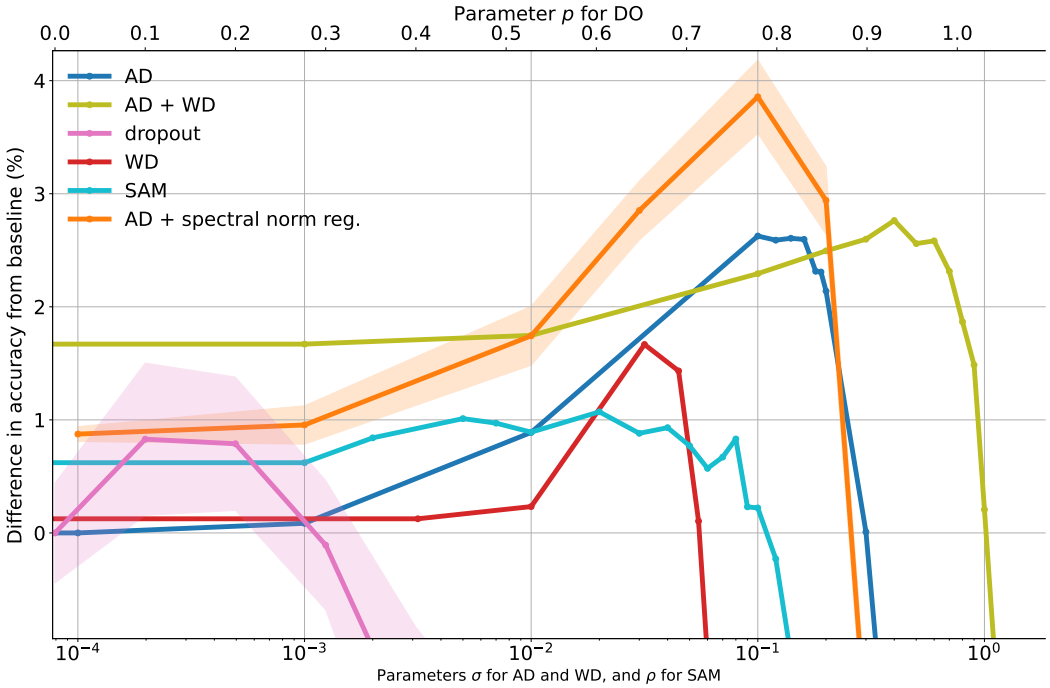


Fig. 7.2.: Comparison of accuracies of different regularizations, **the higher the better**, applied to a 4-layer MLP network trained on the CIFAR-10 dataset. The plots show the evolution of accuracy with varying values of parameter σ for activation decay (AD) and weight decay (WD) and drop rate p for dropout (DO). The first curve (AD) depicts the effect of σ while keeping p at 0.0. The second curve (AD + WD) combines WD with the best parameter $1\text{e}{-3}$ and varying σ for AD. The third curve (DO) illustrates the impact of dropout when σ is 0.0. The fourth curve (WD) depicts the effect of σ when it parameterizes weight decay. The fifth curve (SAM) illustrates the impact of the ρ parameter. The sixth curve (AD + spectral norm reg.) illustrates the combined spectral norm regularization with the parameter value 0.1 for (AD). Shell indicates the standard deviation over 10 runs.

as for CNNs architecture as reported in SAM paper (Foret et al., 2021). We also provide a comparison with a similar approach proposed by Baek et al., 2024 in Appendix D.1.2. Their method applies layer-wise activation regularization across all layers except the last, where weight decay is used. In contrast, our approach requires fewer hyperparameters to tune while achieving comparable results.

We also provide results on MLP-Mixer architecture (Tolstikhin et al., 2021) on ImageNet in the Appendix, showing that our method extends to a bigger dataset and architecture.

Effect of overparameterization on regularization performance This experiment evaluates how AD regularization behaves when varying the number of parameters in the model. We use a 3-layer MLP on CIFAR-10 and adjust the number of hidden features to transition between underparameterized and overparameterized regimes. As shown in Figure 7.3, we observe that the gains from our method are consistent

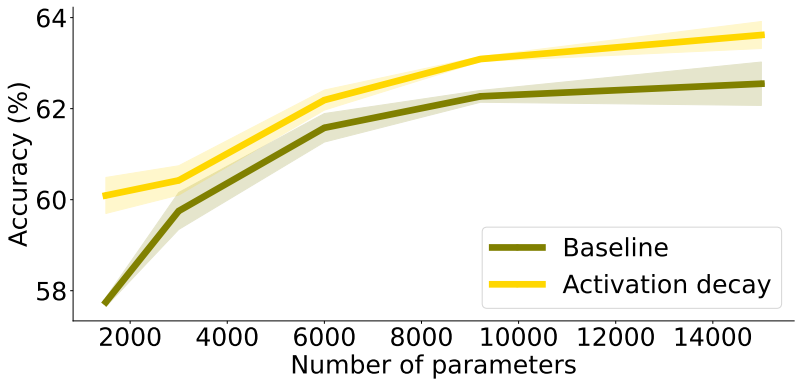


Fig. 7.3.: Accuracy on CIFAR-10 for an MLP with depth 3 and varying numbers of hidden features per layer. Our method with regularization is compared to the baseline with no regularization.

across both regimes. Smoothing the loss leads to noticeable improvements in accuracy, demonstrating the effectiveness of our regularization technique regardless of the model's parameter count.

Tab. 7.3.: Accuracies on validation set for baseline, AD ($\sigma = 0.2$), ASAM ($\rho_{\text{ASAM}} = 2.0$) and AD+SAM, for WideResNet on CIFAR-10. Results were reported after averaging over 3 runs and the standard deviation is 0.03 for all runs.

Configuration	Validation accuracy (%)
Baseline	97.09
ASAM	97.48
AD	97.27
AD + ASAM	**97.54**

Comparison to ASAM with Wide ResNet on CIFAR-10 Table 7.3 presents a comparative analysis of the average accuracies on the validation set achieved by different training configurations using the WideResNet architecture trained on 300 epochs on the CIFAR-10 dataset. The configurations evaluated are: Baseline, the standard training setup without additional optimization techniques, achieved a validation accuracy of 97.09%. Ours, a proposed method utilizing AD with $\sigma = 0.2$, improved the validation accuracy to 97.27%. For experiments involving SAM we use the upgrade Adaptive SAM (ASAM) (Kwon et al., 2021), we adopted the official implementation provided by the authors. To ensure a fair evaluation, we used the best parameter $\rho_{\text{ASAM}} = 2.0$ as specified in their code repository for this particular task which achieved a validation accuracy of 97.48%. AD + ASAM, a combination of the proposed method and SAM yielded the highest validation accuracy of 97.54%. The best values of ASAM and AD parameters were picked by hyperparameter search. AD and ASAM outperform the baseline individually, while the combination yields the best performance. This can be explained because ASAM is designed to minimize

the worst-case ascent sharpness, specifically targeting regions of the loss landscape where the maximum sharpness is reduced (Wen et al., 2022). In contrast, AD focuses on reducing the average sharpness, promoting a smoother and more stable optimization trajectory. This complementary behavior could explain why the two methods combine effectively, leading to enhanced generalization performance.

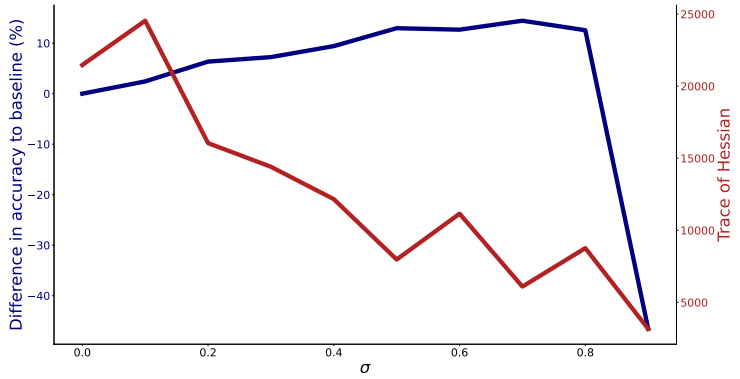


Fig. 7.4.: Accuracy difference and Hessian trace of the loss for varying σ, with 30% label noise on CIFAR-10 using ResNet-56.

Classification with label noise with ResNet-56 on CIFAR-10 We evaluate the effect of 30% label noise effect on CIFAR-10 classification using ResNet-56, varying the parameter σ in AD. Figure 7.4 shows the accuracy difference from a baseline model and the trace of the Hessian of the loss w.r.t all parameters, plotted against σ. The Hessian trace, a proxy for sharpness, indicates sharper minima and poorer generalization under label noise. As σ increases, accuracy initially improves but declines as the Hessian trace rises, aligning with findings from Foret et al. (2021) and Baek et al. (2024), where SAM (Sharpness-Aware Minimization) improves regularization under label noise. Our AD method enhances noise robustness by controlling sharpness as σ increases, improving accuracy without the computational cost of SAM. This makes AD an efficient alternative for handling noisy labels for free. This experiment further reinforced the link between flat minima and label noise robustness. As Lipschitz continuity is tied to generalization properties (Bartlett et al., 2017) and robustness tsuzuku2018lipschitz, flat minima also provides robustness.

7.2.5 Experiment with large language models

We propose to tackle the problem of multi-task learning with large language models (LLMs) using our Activation Decay (AD) method.

Multi-task learning (MTL) Zhang et al., 2021b is a paradigm in machine learning where a model is trained to perform multiple tasks simultaneously, leveraging shared representations across tasks. MTL also offers the benefit of reducing computational

costs and latency, enabling the model to handle multiple tasks in a single inference step, rather than performing separate inferences for each task.

With the advent of large-scale pretrained models, multi-task learning has become increasingly popular in NLP. Pretrained on large corpora, these models can capture a wide range of patterns and dependencies in text data. Leveraging this general knowledge during fine-tuning while specializing in specific tasks leads to new state-of-the-art performances on a variety of downstream tasks. This task-specific adaptation during fine-tuning, is needed for earlier Large Language Models (LLMs) as BERT Devlin et al., 2019 and RoBERTa Liu et al., 2019 and has recently been reduced to few-shot learning with models like GPT-3 Brown et al., 2020 and T5 Raffel et al., 2020 and even extended to zero-shot settings by Sanh et al., 2022; Wei et al., 2022a in the context of very large models.

However, fine-tuning multiple tasks can lead to overfitting on individual tasks, which can degrade the model's performance on other tasks, calling for specific design choices, adding extra task-specific parameters (adapters), or using specific prompts (see Stickland and Murray, 2019; Wang et al., 2023). Our AD method promotes flat minima to improve generalization across tasks. The goal is to prevent overfitting to any specific task while retaining the benefits of pretraining on various tasks. Our flat minima regularization helps preserve the generalization capabilities of the pretrained model, ensuring robust performance across all tasks during fine-tuning. As detailed in Section 2.3.2 flat minima and generalization are often tied together and the experimental results presented in Tables D.1.4 and 7.5 show our regularization helps conserve performance across diverse tasks by promoting smooth optimization landscapes during fine-tuning.

Fine tuning on distinct tasks In this experiment, we evaluate the performance of a multi-task NLP model using the RoBERTa (Liu et al., 2019), BERT (Devlin et al., 2019), and T5 (Raffel et al., 2020) architectures to handle distinct tasks. The multi-task setup allows the model to process these tasks simultaneously, optimizing latency and resource consumption. We experimentally show that in the context of fine-tuning and few-shot learning, our Activation Decay (AD) method helps LLMs to generalize better across tasks, leading to improved performance compared to the baseline models.

Each task is described as follows: (i) Sentiment Analysis: a binary classification task (positive/negative) on the IMDb dataset, using classification accuracy as the metric; (ii) NER: named entity recognition on the Snips and CoNLL datasets, evaluated using F1-score, precision, and recall metrics; (iii) Intent Classification: intent detection on the Snips dataset, evaluated with classification accuracy; (iv) Entailment Clas-

sification (SNLI): a binary classification task predicting whether a sentence entails another, based on the SNLI dataset, with classification accuracy as the metric; (v) POS Tagging: part-of-speech tagging on the CoNLL dataset, evaluated using F1-score, precision, and recall metrics; (vi) Query Correctness: a binary classification task assessing the correctness of queries, evaluated with classification accuracy.

The backbone models used for all tasks are BERT (bert-base) and RoBERTa (roberta-base), with dropout probability set to 0. The model's performance is evaluated using a custom smoothed loss function with $\sigma = 0.05$, and the results are compared with SAM regularization at different ρ values. Weight decay is present by default in the training configuration of the backbone. We use standard training configuration from HuggingFace corresponding models and trainers.

Tab. 7.4.: Evaluation results for BERT baseline with DO ($p = 0.1$), SAM ($\rho = 0.01$), and AD ($\sigma = 0.05$) on 7 tasks.

Metric	DO	SAM	AD
Sentiment Evaluation			
Classification Accuracy (%)	76.72	76.54	**77.08**
NER Evaluation			
Snips F1 Score (%)	78.33	69.67	**80.90**
Snips Precision (%)	73.69	64.11	**76.20**
Snips Recall (%)	83.59	76.28	**86.21**
Intent Evaluation			
Classification Accuracy (%)	98.04	98.19	**98.49**
Entailment SNLI Evaluation			
Classification Accuracy (%)	87.96	**89.39**	88.88
CoNLL NER Evaluation			
Seqeval F1 Score (%)	64.43	61.01	**65.94**
Seqeval Precision (%)	61.87	61.48	**64.11**
Seqeval Recall (%)	67.20	60.55	**67.87**
CoNLL POS Evaluation			
Seqeval F1 Score (%)	75.95	72.48	**77.89**
Seqeval Precision (%)	74.89	71.59	**76.98**
Seqeval Recall (%)	77.04	73.39	**78.82**
Query Correctness Evaluation			
Classification Accuracy (%)	**69.95**	69.47	69.31

Table 7.4 presents evaluation results for fine-tuning BERT on seven NLP tasks, comparing the baseline model with standard dropout ($p = 0.1$), SAM regularization with standard values ($\rho = 0.01$), and Activation Decay ($\sigma = 0.05$). The best values for SAM were selected from $\rho = 0.01, 0.05, 0.1$, and Activation Decay from $\sigma = 0.01, 0.05, 0.1$. Metrics include classification accuracy, F1-score, precision, and recall. Activation Decay consistently outperforms both the baseline and SAM across most

Tab. 7.5.: Test accuracy results for T5 configurations on the MMMLU dataset, for baseline used with DO ($p = 0.1$), and AD ($\sigma = 0.01$).

Model	DO	AD
T5-large	52.07	**52.95**
T5-base	49.89	**50.25**
T5-small	32.21	**33.49**

tasks. The same results for RoBERTa are reported in Table D.1.3 in the Appendix, showing the efficiency of AD.

Few shots learning on MMMLU dataset We evaluate our models on the Multilingual Massive Multitask Language Understanding (MMMLU) dataset (Hendrycks et al., 2020), which spans 57 diverse topics ranging from elementary to advanced professional subjects. We use the newly updated version published by (OpenAI, 2023), which expanded the dataset to include 14 languages using professional human translators. We use the T5 architecture (Raffel et al., 2020) for our experiments, fine-tuning the small (60 M), base (220 M), and large (770 M) variants of the model over 3 epochs. The results, summarized in Table 7.5, show that our AD with $\sigma = 0.01$ in a multitask setting, consistently outperforms the standard dropout $p = 0.1$ a common baseline for fine-tuning baseline across all model sizes, demonstrating its effectiveness for large models. This result highlights the importance of our approach in improving accuracy, particularly in large-scale multitasking environments. All code and implementation details will be made available upon acceptance of the paper to ensure reproducibility.

7.3 Conclusion

We introduced a spectral norm regularization technique that efficiently constrains the Lipschitz constant of individual layers, including convolutions, using the Gram iteration method. This approach stabilizes training, enhances generalization, and offers computational advantages over existing methods, as demonstrated in our CIFAR-10 and ImageNet-1k experiments.

Furthermore, we showed that Lipschitz regularization, which limits network sensitivity to input perturbations, not only improves robustness but also promotes flatter minima (Theorem 7.2.1). To further reduce Hessian curvature, we introduced activation decay as a deterministic alternative to dropout, which relies on noise injection. Activation decay regularizes critical activations via Gaussian smoothing, effectively flattening minima and enhancing robustness without additional computational cost. Unlike Sharpness-Aware Minimization (SAM), which employs a

min-max optimization framework with higher overhead, activation decay provides a computationally efficient alternative that integrates seamlessly with weight decay and Lipschitz regularization. Empirical results validate its effectiveness in replacing dropout and confirm the proposed formulation.

A known limitation of norm-based sharpness metrics is their sensitivity to simple reparameterizations such as neuron-wise rescaling in ReLU networks, which preserve the function but alter spectral or Frobenius norms. This lack of invariance may obscure the relationship between sharpness and generalization, motivating the exploration of scale-invariant alternatives Gonon et al., 2025.

8 Conclusion

Contents

8.1 Thesis summary

In this thesis, we address the problem of stability in deep learning through the lens of sensitivity analysis from various perspectives: sensitivity to inputs with Lipschitz networks and to parameters with loss curvature. Specifically, we advocate for the use of Lipschitz networks as a means to enhance the stability and robustness of deep learning models. By combining Lipschitz networks with Hessian-bounded loss regularization, this thesis introduces a unified framework for tackling long-standing challenges in stability and robustness. Through spectral norm computation, randomized smoothing, and curvature regularization, we provide both theoretical insights and practical methodologies. Our key contributions can be summarized as follows:

Summary of the contributions:

1. *We developed Gram iteration and its derived algorithms for computing the spectral norm of both convolutional and dense layers in deep learning networks.*
2. *We established a connection between randomized smoothing—a probabilistic method for certifying robustness—and the Lipschitz constant of neural networks. This interaction improves stability, leading to better certificates and reduced variance.*
3. *We proposed a novel method to generate tighter confidence intervals in the context of randomized smoothing, further improving the reliability of robustness estimates.*
4. *We introduced a new regularization term based on the Hessian of the loss function to enhance the stability of neural networks.*

These contributions provide both theoretical and practical advancements in the study of neural network stability and robustness. By improving spectral norm estimation, establishing connections between Lipschitz continuity and randomized smoothing,

refining confidence intervals, and introducing Hessian-based regularization, our work enhances the reliability of robustness guarantees while maintaining computational efficiency. These methods contribute to a better understanding of generalization, adversarial robustness, and certified defenses in deep learning. However, several challenges remain to be addressed in the field of Lipschitz networks, including scaling, attention mechanisms, and Lipschitz-certified robustness for large language models.

8.2 Open problems

Scaling Lipschitz networks Scaling generally improves performance, but Lipschitz networks have yet to match the accuracy and efficiency of standard architectures. Despite this, they offer significant advantages in robustness, stability, and generalization. Unlike traditional networks, Lipschitz models train reliably without requiring tricks such as batch normalization or gradient clipping, making them inherently stable and easier to optimize Meunier et al., 2022.

Lipschitz networks require more extensive parameter tuning than standard networks (Bubeck et al., 2021; Bubeck and Sellke, 2021). Developing foundational models with pre-tuned parameters would simplify their deployment. Indeed, the works of Araujo et al. (2023) and Hu et al. (2023) have shown that adding large feedforward layers after convolutional layers significantly improves performance, as this overparameterizes the network and enhances its capacity. Investigating the scaling laws of robustness in relation to the Lipschitz constant could offer valuable insights for designing scalable Lipschitz networks. Training large Lipschitz networks on extensive datasets in a highly overparameterized regime is a promising direction. Moreover, similar scaling laws apply to data, where the use of synthetic data has proven highly effective in increasing performance (Hu et al., 2023; Hu et al., 2024). However, this implies a significant computational cost and energy consumption, which is not always feasible in practice and goes against the goal of frugality in model use and design.

While scaling could be considered the primary explanation for this gap, the inherent limitations of Lipschitz networks suggest that scaling alone may not be sufficient. Randomized smoothing provides a theoretical framework for defining efficient Lipschitz networks, even if they can only be approximated in practice using Monte Carlo methods. Recently models combining diffusion-based denoisers with classifiers have demonstrated promising results in terms of robustness and accuracy on large datasets such as ImageNet (Carlini et al., 2023), with performance close to standard networks. This highlights a crucial perspective: the existence of such architectures suggests that

the gap can still be bridged by developing innovative designs that balance training stability, performance, and computational efficiency. Further research is needed to understand the trade-offs in these models and to propose new architectures that push the boundaries of Lipschitz network performance. Recent works on by-design Lipschitz networks, such as Hu et al. (2023) and Hu et al. (2024) pave the way and encourage the community to explore this direction, demonstrating significant improvements in scalability and robustness. Another perspective to narrow this gap could be to explore the development of attention mechanisms for Lipschitz networks, as they are a key component in many recent state-of-the-art models.

Lipschitz constraints for attention mechanisms Self-attention mechanisms have become a fundamental component of modern deep learning architectures, particularly in natural language processing (NLP) (Vaswani et al., 2017). However, self-attention relies on interactions between keys and queries, leading to a quadratic transformation of the inputs, which inherently makes it non-Lipschitz. This poses a challenge when designing Lipschitz-constrained attention layers. Modern architectures consistently incorporate residual connections, which suggests a possible workaround: instead of enforcing Lipschitz constraints on self-attention alone, one could consider the Lipschitz behavior of the entire residual transformation, including both the self-attention mechanism and the residual path.

A promising approach in this direction is the framework of convex potential flows introduced by Meunier et al. (2022), which enables Lipschitz residual transformations through carefully chosen discretizations of continuous flows. Extending this idea to self-attention would require designing a convex potential that naturally induces an attention-like transformation while preserving Lipschitz's continuity.

Enhancing randomized smoothing certification In randomized smoothing, the key quantity of interest is the certified radius that guarantees robustness. This radius is determined by two main factors: the Lipschitz constant of the margin function and the margin of the smoothed classifier at a given point. Since the true margin of the smoothed classifier is not directly accessible, it is estimated using confidence intervals on each logit coordinate, from which the margin is then computed. Similarly, the Lipschitz constant of the margin function is typically bounded using the Lipschitz constants of the individual logit coordinates, leading to the expression of the radius $\mathrm{R}_{\mathrm{coord}}$ in (2.9).

A potential improvement to this procedure would be to estimate the margin of the smoothed classifier directly, rather than first estimating the margin of the logits and then deriving the margin of the smoothed classifier. Developing a method for direct estimation could reduce variance, leading to tighter confidence intervals

and ultimately improving the certified radius. The same principle applies to the Lipschitz constant of the margin function: instead of relying on bounds derived from individual logit coordinates, estimating it directly from the full margin function could provide a more accurate and potentially tighter bound. This would involve leveraging the first expression of the radius in (2.8) and incorporating smooth margins.

The main challenge in implementing this approach lies in adapting it to the radius defined by the composition of the Gaussian quantile function with the smoothed classifier. Indeed, the composed function is not bounded, which necessitates a careful analysis to construct a reliable confidence interval.

Certified robustness for large language models Certified robustness remains an open challenge for large language models (LLMs), as adversarial vulnerabilities have been observed in these architectures (Huang et al., 2019; Jia and Liang, 2019). While certified defenses have been well studied in computer vision, their extension to NLP is less straightforward due to the discrete nature of text inputs and the difficulty of defining meaningful perturbations. Developing robust certification methods for LLMs that account for adversarial attacks while maintaining linguistic coherence remains an important research direction.

A Spectral norm computation

A.0.1 Toeplitz matrix

Let $\mathbf{W}_{\text{toep}} \in \mathbb{R}^{c_{\text{out}}n^2 \times c_{\text{in}}n^2}$ be a $c_{\text{out}} \times c_{\text{in}}$ block matrix:

$$\mathbf{W}_{\text{toep}} = \begin{pmatrix} \mathbf{W}_{\text{toep}_{1,1,:,:}} & \mathbf{W}_{\text{toep}_{1,2,:,:}} & \cdots & \mathbf{W}_{\text{toep}_{1,c_{\text{in}},:,:}} \\ \mathbf{W}_{\text{toep}_{2,1,:,:}} & \mathbf{W}_{\text{toep}_{2,2,:,:}} & \cdots & \mathbf{W}_{\text{toep}_{2,c_{\text{in}},:,:}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_{\text{toep}_{c_{\text{out}},1,:,:}} & \mathbf{W}_{\text{toep}_{c_{\text{out}},2,:,:}} & \cdots & \mathbf{W}_{\text{toep}_{c_{\text{out}},c_{\text{in}},:,:}} \end{pmatrix},$$

where each $\mathbf{W}_{\text{toep}_{j,i,:,:}}$ is a $n^2 \times n^2$ banded doubly Toeplitz matrix formed with kernel $K_{j,i}$. The matrix $\mathbf{W}_{\text{toep}_{j,i,:,:}}$ is represented in Equation (A.1), where we define:

$$\text{toep}(K_{j,i,k_1}) = \begin{pmatrix} K_{j,i,k_1,\lfloor \frac{k}{2} \rfloor} & \cdots & K_{j,i,k_1,1} & 0 & \cdots & 0 \\ \vdots & K_{j,i,k_1,\lfloor \frac{k}{2} \rfloor} & \ddots & \ddots & \ddots & \vdots \\ K_{j,i,k_1,k} & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & K_{j,i,k_1,1} \\ \vdots & \ddots & \ddots & \ddots & K_{j,i,k_1,\lfloor \frac{k}{2} \rfloor} & \vdots \\ 0 & \cdots & 0 & K_{j,i,k_1,k} & \cdots & K_{j,i,k_1,\lfloor \frac{k}{2} \rfloor} \end{pmatrix}$$

A.0.2 Circulant matrix

In the same manner, as for zero padding with matrix $\mathbf{W}_{\text{toep}}$, $\mathbf{W}_{\text{circ}}$ can also be represented as a $c_{\text{out}} \times c_{\text{in}}$ block matrix of doubly circulant matrices. $\mathbf{W}_{\text{circ}j,i,:,:}$ is fully determined by the kernel K as:

$$\mathbf{W}_{\text{circ}j,i,:,:} = \begin{pmatrix} \text{circ}(K_{j,i,1,:}) & \text{circ}(K_{j,i,2,:}) & \cdots & \text{circ}(K_{j,i,n,:}) \\ \text{circ}(K_{j,i,n,:}) & \text{circ}(K_{j,i,1,:}) & \ddots & \vdots \\ \vdots & \ddots & \text{circ}(K_{j,i,1,:}) & \text{circ}(K_{j,i,2,:}) \\ \text{circ}(K_{j,i,2,:}) & \cdots & \text{circ}(K_{j,i,n,:}) & \text{circ}(K_{j,i,1,:}) \end{pmatrix}.$$

We denote $\mathbf{W}_{\text{circ}j,i,:,:} = \text{blkcirc}(K_{j,i,1,:}, \ldots, K_{j,i,n,:})$.

$$\mathbf{W}_{\mathrm{toep}\,j,i,:,:} = \begin{pmatrix} \mathrm{toep}(K_{j,i,\lfloor \frac{k}{2} \rfloor}) & \cdots & \mathrm{toep}(K_{j,i,1}) & 0 & \cdots & 0 \\ \vdots & \mathrm{toep}(K_{j,i,\lfloor \frac{k}{2} \rfloor+1}) & \ddots & \ddots & \ddots & \vdots \\ \mathrm{toep}(K_{j,i,k}) & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \mathrm{toep}(K_{j,i,1}) \\ \vdots & \ddots & \ddots & \ddots & \mathrm{toep}(K_{j,i,\lfloor \frac{k}{2} \rfloor}) & \vdots \\ 0 & \cdots & 0 & \mathrm{toep}(K_{j,i,k}) & \cdots & \mathrm{toep}(K_{j,i,\lfloor \frac{k}{2} \rfloor}) \end{pmatrix} \tag{A.1}$$

A.0.3 Proof of Theorem 3.2.4

Proof. Closed form. By induction, for $t \geq 2$,

$$\mathbf{W}^{(t)} = (\mathbf{W}^{*}\mathbf{W})^{2^{t-2}} =: \mathbf{A}^{k}, \qquad k := 2^{t-2},$$

so

$$s_t = \|\mathbf{W}^{(t)}\|^{2^{1-t}} = \|\mathbf{A}^{k}\|^{1/(2k)}.$$

Convergence and upper bound. Gelfand's formula gives $\lim_{k\to\infty} \|\mathbf{A}^{k}\|^{1/k} = \rho(\mathbf{A}) = \sigma_1(\mathbf{W})^2$, hence $s_t \to \sigma_1(\mathbf{W})$. Since $\rho(\mathbf{X}) \leq \|\mathbf{X}\|$ for any norm, $\sigma_1(\mathbf{W})^{2^{t-1}} = \rho(\mathbf{A})^{k} \leq \|\mathbf{A}^{k}\|$, and raising to 2^{1-t} yields $s_t \geq \sigma_1(\mathbf{W})$.

Rate (general norm). By norm equivalence in finite dimension, $\exists\, 0 < m \leq M < \infty$ such that $m\|\mathbf{X}\|_2 \leq \|\mathbf{X}\| \leq M\|\mathbf{X}\|_2$. For $\mathbf{A} \geq 0$, $\|\mathbf{A}^{k}\|_2 = \rho(\mathbf{A})^{k}$. Thus

$$\sigma_1(\mathbf{W})\, e^{\frac{\log m}{2^{t-1}}} \leq s_t \leq \sigma_1(\mathbf{W})\, e^{\frac{\log M}{2^{t-1}}}$$

and $s_t - \sigma_1(\mathbf{W}) = O(2^{-t})$.

Frobenius readout. Suppose the final norm is $\|\cdot\|_F$. Let the eigenvalues of $\mathbf{A}$ be $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ and set $q := \sqrt{\lambda_2/\lambda_1} = \sigma_2(\mathbf{W})/\sigma_1(\mathbf{W}) \in (0,1)$. Then

$$\|\mathbf{A}^{k}\|_F^2 = \sum_i \lambda_i^{2k} = \lambda_1^{2k}\left(1 + \sum_{i\geq 2} (\lambda_i/\lambda_1)^{2k}\right) = \lambda_1^{2k}\,(1 + S_k),$$

with $0 \leq S_k \leq (r-1)\, q^{4k}$ ($r = \mathrm{rank}\,\mathbf{A}$) and $S_k \geq q^{4k}$ if $\lambda_2 > 0$. Hence

$$s_t = \|\mathbf{A}^{k}\|_F^{1/(2k)} = \sigma_1(\mathbf{W})\,(1 + S_k)^{1/(4k)}, \qquad e_t = \sigma_1(\mathbf{W})\big[(1 + S_k)^{1/(4k)} - 1\big].$$

Using $\log(1+u) = u + O(u^2)$ and $4k = 2^t$,

$$e_t = \sigma_1(\mathbf{W})\,\frac{S_k}{4k}\,(1 + o(1)).$$

Consequently, for some constants $0 < c \le C < \infty$,

$$c\,\frac{q^{2^t}}{2^t} \;\le\; e_t \;\le\; C\,\frac{q^{2^t}}{2^t} \qquad (t \to \infty).$$

Given any $\varepsilon > 0$, since $\left(\frac{q}{q+\varepsilon}\right)^{2^t} 2^t \to 0$, there exist C_ε, T with $e_t \le C_\varepsilon\,(q+\varepsilon)^{2^t}$ for all $t \ge T$: this is K-supergeometric with $K = q$.
For the "order", write $\log e_t = \log c + 2^t \log q - t \log 2 + o(1)$ (with some $c > 0$), so

$$p_t := \frac{\log e_{t+1}}{\log e_t} = \frac{\log c + 2^{t+1}\log q - (t+1)\log 2 + o(1)}{\log c + 2^t \log q - t \log 2 + o(1)} = 2 - \Theta\Big(\frac{t}{2^t}\Big),$$

hence $p_t \uparrow 2$.
(Not R-quadratic.) From $e_t = \Theta(q^{2^t}/2^t)$ one gets $e_{t+1}/e_t^2 \sim (2^{t-1}/\sigma_1(\mathbf{W})) \to \infty$, so no inequality $e_{t+1} \le K e_t^2$ can hold for all large t.
Operator 2-norm. If the final norm is $\|\cdot\|_2$ then $\|\mathbf{A}^k\|_2 = \rho(\mathbf{A})^k$ and $s_t \equiv \sigma_1(\mathbf{W})$ for $t \ge 2$.

A.0.4 Proof of Theorem 3.6.1

Proof.

$$\mathbf{W}_{\mathrm{toep}_{j,i,k_2 l_2,k_1 l_1}} = K_{i,j,k_1-k_2+k-1,l_1-l_2+k-1}$$

For $\mathbf{W}_{\mathrm{toep}}{}^{(1)} \in \mathbb{R}^{n^2 c_{\mathrm{in}} \times n^2 c_{\mathrm{in}}}$, the first Gram iterate, $1 \le i_1, i_2 \le c_{\mathrm{in}}$,

$$\mathbf{W}^{(1)}_{\mathrm{toep}_{i_1,i_2,k_1 l_1,k_2 l_2}} = \left(\sum_{j=1}^{c_{\mathrm{out}}} K_{j,i_1} \star K_{j,i_2}\right)_{k_2-k_1,l_2-l_1}$$

It was first derived in Prach and Lampert (2022).
We define the Gram iterate for the filter K, $K^{(t+1)}_{i_1,i_2} = \sum_{j=1}^{c_{\mathrm{out}}} K^{(t)}_{j,i_1} \star K^{(t)}_{j,i_2}$, where convolution is defined with maximal non-trivial padding. We pad $K^{(t)}$ with zeros corresponding to the current spatial size of the kernel of $K^{(t)}$ minus one.
We can extend the previous result for the t-th iterate of Gram:

$$\mathbf{W}^{(t+1)}_{\mathrm{toep}_{i_1,i_2,k_1 l_1,k_2 l_2}} = \left(\sum_{j=1}^{c_{\mathrm{out}}} K^{(t)}_{j,i_1} \star K^{(t)}_{j,i_2}\right)_{k_2-k_1,l_2-l_1}. \tag{A.2}$$

For the norm $\|\,.\,\|_\infty$, we use the tight bound which simplifies to:

$$\|\mathbf{W}_{\text{toep}}{}^{(t+1)}\|_\infty = \max_{i_2,k_2,l_2} \sum_{i_1,k_1,l_1} \left| \sum_{j=1}^{c_{\text{out}}} K_{j,i_1}^{(t)} \star K_{j,i_2}^{(t)} \right|_{k_2-k_1,l_2-l_1}$$
$$\leq \max_{i_2} \sum_{i_1,k',l'} \left| \sum_{j=1}^{c_{\text{out}}} K_{j,i_1}^{(t)} \star K_{j,i_2}^{(t)} \right|_{k',l'} .$$

For the norm $\|\,.\,\|_{\text{F}}$, it gives:

$$\|\mathbf{W}_{\text{toep}}{}^{(t+1)}\|_{\text{F}}^2 = \sum_{i_1,i_2,k_1,k_2,l_1,l_2} \left| \sum_{j=1}^{c_{\text{out}}} K_{j,i_1}^{(t)} \star K_{j,i_2}^{(t)} \right|_{k_2-k_1,l_2-l_1}^2$$
$$\leq k^2 \sum_{i_1,i_2,k',l'} \left| \sum_{j=1}^{c_{\text{out}}} K_{j,i_1}^{(t)} \star K_{j,i_2}^{(t)} \right|_{k',l'}^2 .$$

A.0.5 Proof of Theorem 3.7.3

Theorem A.0.1 (Adapted from Theorem 1 of Pfister and Bresler (2019))**.**
Let $\gamma : \mathbb{R}^2 \to \mathbb{C}$ be a trigonometric polynomial of degree $d = \lfloor k/2 \rfloor$ defined by coefficients $\Gamma \in \mathbb{C}^{k\times k}$:

$$\gamma(w_1, w_2) = \sum_{k_1=0}^{2d} \sum_{k_2=0}^{2d} \Gamma_{k_1,k_2}\, e^{\mathbf{i}(w_1(k_1-d))} e^{\mathbf{i}(w_2(k_2-d))} .$$

Let Ω_n be the set of n equidistant sampling points on $[0, 2\pi]$: $\Omega_n = \left\{\omega_l = \frac{2\pi l}{n} \mid l \in \{1, \ldots, n\}\right\}$. Then, for $\alpha = \frac{2d}{n}$, we have:

$$\max_{w_1,w_2\in[0,2\pi]^2} |\gamma(w_1, w_2)| \leq (1-\alpha)^{-1} \max_{w_1',w_2'\in\Omega_n^2} \left|\gamma(w_1', w_2')\right| .$$

We define $\mathbf{E} \in \mathbb{C}^{c_{\text{out}}\times c_{\text{in}}}$ as the spectral density matrix of the filter K:

$$\mathbf{E}(w_1, w_2) = \sum_{k_1=0}^{k-1} \sum_{k_2=0}^{k-1} e^{-\mathbf{i}k_1 w_1}\, e^{-\mathbf{i}k_2 w_2}\, K_{:,:,k_1,k_2}. \tag{A.3}$$

The spectral density matrix $\mathbf{E}$ corresponds to the Discrete Time Fourier Transform (DTFT) of the filter K, where $K_{:,:,k_1,k_2}$ represents the coefficients of the convolution kernel.

Although $\mathbf{E}$ is directly defined in terms of K, it provides key insights into the spectral properties of the Toeplitz matrix $\mathbf{W}_{\text{toep}}$, as $\mathbf{W}_{\text{toep}}$ is constructed based on K. Iteratively, $\mathbf{E}^{(t)}$ denotes the t^{th} Gram iterate of the matrix $\mathbf{E}$, which is used to analyze the spectral behavior of $\mathbf{W}_{\text{toep}}$.

Using Theorem A.0.1, we can bound the Gram iterate maximum norm over $[0, 2\pi]$ of $\mathbf{E}$ with the maximum taken over uniform sampled points.

Lemma A.0.2.

(Inequality between the maximum of spectral norm density and density uniformly sampled for Gram iterates)

For $n_0 \geq 2^t\lfloor \frac{k}{2} \rfloor + 1$ *sampling points,* $\alpha = \frac{2^t\lfloor \frac{k}{2} \rfloor}{n_0}$*:*

$$\max_{w_1,w_2\in[0,2\pi]} \|\mathbf{E}^{(t)}(w_1, w_2)\|_{\mathrm{F}}^2 \leq (1-\alpha)^{-1} \max_{1\leq u,v\leq n_0} \|\mathbf{D}^{\downarrow(t)}_{u,v}\|_{\mathrm{F}}^2$$

Proof. We define the trigonometric polynomial of degree $2^t\lfloor \frac{k}{2} \rfloor$: $\mathbf{P} = \|\mathbf{E}^{(t)}(\,.\,,\,.\,)\|_{\mathrm{F}}^2$.

$$\begin{aligned}
&\max_{w_1,w_2\in[0,2\pi]} \|\mathbf{E}^{(t)}(w_1, w_2)\|_{\mathrm{F}}^2 \\
&= \max_{w_1,w_2\in[0,2\pi]^2} \mathbf{P}(w_1, w_2) \\
&\leq (1-\alpha)^{-1} \max_{w_1,w_2\in\Omega^2_{n_0}} \mathbf{P}(w_1, w_2) \quad \text{applying Theorem A.0.1} \\
&= (1-\alpha)^{-1} \max_{1\leq u,v\leq n_0} \|\mathbf{D}^{\downarrow(t)}_{u,v}\|_{\mathrm{F}}^2 \,.
\end{aligned}$$

Now we can prove Theorem 3.7.3.

Proof. We can cast $\max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}{}^{(t)}\|_{\mathrm{F}}^2$ as the maximum over Ω_n^2 of $\|\mathbf{E}^{(t)}(w_1, w_2)\|_{\mathrm{F}}^2$.

$$\begin{aligned}
\max_{1\leq u,v\leq n} \|\mathbf{D}^{(t)}_{u,v}\|_{\mathrm{F}}^2 &= \max_{w_1,w_2\in\Omega_n^2} \|\mathbf{E}^{(t)}(w_1, w_2)\|_F^2 \\
&\leq \max_{w_1,w_2\in[0,2\pi]^2} \|\mathbf{E}^{(t)}(w_1, w_2)\|_{\mathrm{F}}^2
\end{aligned}$$

Using Lemma A.0.2:

$$\max_{1\leq u,v\leq n} \|\mathbf{D}^{(t)}_{u,v}\|_{\mathrm{F}}^2 \leq (1-\alpha)^{-1} \max_{1\leq u,v\leq n_0} \|\mathbf{D}^{\downarrow(t)}_{u,v}\|_{\mathrm{F}}^2 \,.$$

Finally, using that

$$\sigma_1(\mathbf{W}_{\mathrm{circ}}) = \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}\|_2 \leq \max_{1\leq u,v\leq n} \|\mathbf{D}^{(t)}_{u,v}\|_{\mathrm{F}}^{2^{1-t}} \,,$$

we have

$$\sigma_1(\mathbf{W}_{\mathrm{circ}}) \leq \left(\frac{1}{1-\alpha}\right)^{2^{-t}} \max_{1\leq u,v\leq n_0} \|\mathbf{D}^{\downarrow(t)}_{u,v}\|_{\mathrm{F}}^{2^{1-t}} \,.$$

A.0.6 Proof of Theorem 3.7.1

Proof. We use the spectral density matrix $\mathbf{E}$ defined in Equation A.3. Lemma 4 of Yi (2020) shows that the spectral norm of the Toeplitz matrix is bounded by the spectral norm of its density matrix: $\sigma_1(\mathbf{W}_{\text{toep}}) \leq \sigma_1(\mathbf{E})$. Then,

$$\sigma_1(\mathbf{W}_{\text{toep}})^{2^t} \leq \sigma_1(\mathbf{E})^{2^t} \leq \|\mathbf{E}^{(t)}\|_{\text{F}}^2 .$$

Using Lemma A.0.2 with n sampling points:

$$\max_{w_1,w_2\in[0,2\pi]} \|\mathbf{E}^{(t)}(w_1,w_2)\|_{\text{F}}^2 \leq (1-\alpha)^{-1} \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^2 .$$

Finally,

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \left(\frac{1}{1-\alpha}\right)^{2^{-t}} \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^{2^{1-t}} .$$

A.0.7 Proof of Proposition 3.7.2

Proof. As in the previous proof, Lemma 4 of Yi (2020) shows that the spectral norm of the Toeplitz matrix $\mathbf{W}_{\text{toep}}$ is bounded by the spectral norm of its density matrix $\mathbf{E}$, defined in Eq. (A.3):

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \sigma_1(\mathbf{E}).$$

Using Lemma A.0.2 with n sampling points, we have:

$$\max_{w_1,w_2\in[0,2\pi]} \|\mathbf{E}^{(t)}(w_1,w_2)\|_{\text{F}}^2 \leq (1-\alpha)^{-1} \max_{1\leq u,v\leq n} \|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^2.$$

In the case where $c = c_{\text{out}} = c_{\text{in}}$, for orthogonal circular convolution, we have $\|\mathbf{D}_{u,v}^{(t)}\|_{\text{F}}^2 = c^2$. Thus:

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq \left(\frac{c}{1-\alpha}\right)^{2^{-t}} .$$

Taking the limits $n \to \infty$ and $t \to \infty$, we obtain:

$$\sigma_1(\mathbf{W}_{\text{toep}}) \leq 1.$$

This completes the proof.

A.0.8 Additional experiments

Tab. A.0.1.: Numerical values associated to Figure 3.4: error ratios and computational times of methods for spectral norm computation. Error ratio is defined as $\sigma_{1\,\text{method}}/\sigma_{1\,\text{ref}} - 1$.

Method	Error ratio	Computational time (s)
Reference	0	1.47e+0
PI (10 iters)	-3.27e-2 ± 3.55e-3	1.12e-3
PI (100 iters)	-1.41e-3 ± 1.75e-3	1.03e-2
PI (1000 iters)	-2.43e-6 ± 4.01e-6	1.04e-1
Ours GI (10 iters)	6.75e-6	1.83e-3
Ours GI (11 iters)	4.73e-8	1.99e-3
Ours GI (12 iters)	4.33e-12	2.13e-3

Tab. A.0.2.: Times and difference to reference, for different $p \times p$ matrices, averaged 10 times. Error is defined as $\sigma_{\text{method}} - \sigma_{\text{ref}}$. To have comparable time and precision, 100 iterations were taken for Power iteration and 10 for Gram iteration. Gram iteration is overall more competitive than Power iteration for all matrix dimensions considered.

Dimension p	PI time (s)	GI time (s)	PI error	GI error	Ref
10	0.0093	0.0012	0.0006	0	5.28
100	0.0083	0.0010	-0.0231	0	19.31
1000	0.0093	0.0017	-0.0426	0.0005	62.95
10000	0.016	0.0019	-0.6004	0.0037	199.77
50000	0.017	0.0040	-1.70	0.1	447.21

Tab. A.0.3.: Comparison of Lipschitz bounds for all convolutional layers ResNet18, reference for real Lipschitz bound is given by Ryu et al. (2019) method. We observe that our method gives close value to Sedghi et al. (2019) up to two decimal digits while being significantly faster. We remark that convolution filter $512 \times 512 \times 1 \times 1$ Singla and Feizi (2021a) value 2.03 underestimates reference value 2.05.

Filter Shape	Lipschitz Bound					Running Time (s)				
	Ours	Singla2021	Araujo2021	Sedghi2019	Ryu2019	Ours	Singla2021	Araujo2021	Sedghi2019	Ryu2019
64 × 3 × 7 × 7	15.91	28.89	22.25	15.91	15.91	0.0024	0.0507	0.0003	18.06	0.7579
64 × 64 × 3 × 3	6.00	9.32	17.59	6.00	6.00	0.0014	0.0264	0.0003	7.47	1.07
64 × 64 × 3 × 3	5.34	6.28	15.31	5.34	5.33	0.0015	0.025	0.0012	7.29	1.07
64 × 64 × 3 × 3	7.00	8.71	16.46	7.00	7	0.0011	0.026	0.0003	7.27	1.08
64 × 64 × 3 × 3	3.82	5.40	13.74	3.81	3.82	0.0014	0.025	0.0003	7.19	1.08
128 × 64 × 3 × 3	4.71	5.98	16.35	4.71	4.71	0.0014	0.026	0.0012	9	0.473
128 × 128 × 3 × 3	5.72	7.21	23.58	5.72	5.72	0.0011	0.026	0.0013	4.85	0.78
128 × 64× 1 × 1	1.91	1.91	6.39	1.91	1.91	0.0014	0.0333	0.0001	9.25	0.042
128 × 128 × 3 × 3	4.39	6.78	21.93	4.39	4.41	0.0011	0.026	0.0003	4.844	0.78
128 × 128 × 3 × 3	4.89	7.57	19.79	4.89	4.88	0.0014	0.027	0.0004	4.78	0.78
256 × 128 × 3 × 3	7.39	8.45	35.86	7.39	7.39	0.0014	0.026	0.0017	5.36	0.46
256 × 256 × 3 × 3	6.58	8.03	37.98	6.58	6.58	0.0014	0.026	0.0025	3.46	0.85
256 × 128× 1 × 1	1.21	1.21	5.97	1.21	1.21	0.0014	0.036	0.0001	5.96	0.04
256 × 256 × 3 × 3	6.36	7.57	30.07	6.36	6.35	0.001	0.026	0.0004	3.48	0.85
256 × 256 × 3 × 3	7.68	9.17	29.19	7.68	7.67	0.0011	0.027	0.0004	3.48	0.85
512 × 256 × 3 × 3	9.99	11	46.24	9.99	9.92	0.0011	0.026	0.004	3.55	0.45
512 × 512 × 3 × 3	9.09	10.45	47.59	9.09	9.04	0.0014	0.026	0.0001	2.8	0.79
512 × 256 × 1 × 1	2.05	2.03	11.87	2.05	2.05	0.0012	0.024	0.0001	3.77	0.052
512 × 512 × 3 × 3	17.60	18.37	59.04	17.60	17.5	0.001	0.027	0.0005	2.79	0.79
512 × 512 × 3 × 3	7.48	7.6	57.33	7.48	7.43	0.0014	0.027	0.0006	2.83	0.79

B Lipschitz networks

B.1 Computation of Product Upper Bound (PUB) for ResNet architectures

The Product Upper Bound (PUB) for the Lipschitz constant of a ResNet model is computed as the product of the Lipschitz constants of its individual layers. Below, we summarize how the Lipschitz constant is estimated for different components in the ResNet architecture.

For convolutional layers (Conv2D), the Lipschitz constant is the spectral norm of the layer, which corresponds to the largest singular value of the weight matrix. This is computed via power iteration applied directly to the convolution operator as in Ryu et al. (2019). The iteration alternates between forward and transposed convolutions until convergence.

For fully connected (dense) layers, the Lipschitz constant is also the spectral norm of the weight matrix. Power iteration is similarly applied, where random vectors are iteratively multiplied by the weight matrix and its transpose to approximate the largest singular value.

Batch normalization layers (BatchNorm) are treated as affine transformations during inference. The Lipschitz constant is computed as:

$$\mathrm{L}_{\mathrm{BN}} = \max_i \left| \frac{\gamma_i}{\sqrt{\sigma_i^2 + \epsilon}} \right|,$$

where γ_i is the scale parameter, σ_i^2 is the running variance, and ϵ is a small stability constant. This ensures the Lipschitz constant accurately reflects the effect of batch normalization during evaluation.

Pooling layers, including max pooling and average pooling, are conservatively assumed to have a Lipschitz constant of 1. Max pooling performs a maximum operation over the input, which does not amplify the norm, and average pooling is a linear operation with bounded norm.

Residual connections involve summing the output of a subnetwork with its input. The Lipschitz constant of a residual block is given by:

$$\mathrm{L}_{\text{residual}} \leq L_{\text{main}} + 1,$$

where L_{main} is the Lipschitz constant of the main path (e.g., convolutional layers). The identity mapping has a Lipschitz constant of 1, and the addition operation preserves the norm.

The overall PUB for a ResNet model is computed by multiplying the Lipschitz constants of all layers, taking into account the components described above. This product provides a conservative upper bound for the Lipschitz constant of the network, which is crucial for robust certification and stability analysis.

B.2 Proof of Theorem 4.5.1

Proof. Using Theorem 1 of Araujo et al., 2023, we have to show that $\mathbf{W}^\top\mathbf{W} - \mathbb{T} \preceq 0$, for $t \geq 1$.
The case $t = 1$ comes directly from Theorem 3 of Araujo et al., 2023.
For $t \geq 1$, using Theorem 3 of Araujo et al., 2023 on $\mathbf{W}^{(t)}$,

$$\mathbf{W}^{(t)\top}\mathbf{W}^{(t)} - \operatorname{diag}\left(\sum_j \left|\mathbf{W}^{(t)\top}\mathbf{W}^{(t)}\right|_{ij} \frac{q_j}{q_i}\right) \preceq 0$$

$$(\mathbf{W}^\top\mathbf{W})^{2^t} - \operatorname{diag}\left(\sum_j \left|\mathbf{W}^{(t+1)}\right|_{ij} \frac{q_j}{q_i}\right) \preceq 0\ .$$

Using Zhang, 2011, Theorem 7.9, p. 210, we have that for $A, B \succeq 0$, $A^2 \preceq B^2 \implies A \preceq B$:

$$\mathbf{W}^\top\mathbf{W} - \operatorname{diag}\left(\sum_j \left|\mathbf{W}^{(t+1)}\right|_{ij} \frac{q_j}{q_i}\right)^{2^{-t}} \preceq 0\ .$$

B.3 Proof of Theorem 4.6.1 Lipschitz bound for the Weierstrass transform with based Lipschitz continuity

Stein's lemma can be easily extended to $h : \mathbb{R}^d \mapsto \mathbb{R}^c$. We note $\frac{\partial}{\partial x}\tilde{h}(x)$ the Jacobian matrix of $\tilde{h}(x)$.

Lemma B.3.1.
Let $\sigma > 0$, let $h : \mathbb{R}^d \mapsto \mathbb{R}^c$ be measurable, and let $\tilde{h}(x) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[h(x+\delta)]$. Then $\tilde{h}$ is differentiable, and moreover,

$$\frac{\partial \tilde{h}(x)}{\partial x} = \frac{1}{\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[\delta h(x+\delta)^\top] \ .$$

Proof. We are not going to prove differentiability as it is the same argument as in the proof of Lemma 2.1.4, see (Stein, 1981).

$$\begin{aligned}
\frac{\partial}{\partial x}\tilde{h}(x) &= \frac{\partial}{\partial x}\left(\frac{1}{(2\pi\sigma^2)^{d/2}}\int_{\mathbb{R}^d} h(t)\exp\left(-\frac{1}{2\sigma^2}\|x-t\|^2\right)dt\right) \\
&= \frac{1}{(2\pi\sigma^2)^{d/2}}\int_{\mathbb{R}^d}\frac{\partial}{\partial x}\left(\exp\left(-\frac{1}{2\sigma^2}\|x-t\|^2\right)\right)h(t)^\top dt \\
&= \frac{1}{(2\pi\sigma^2)^{d/2}}\int_{\mathbb{R}^d}\frac{\partial}{\partial x}\left(\exp\left(-\frac{1}{2\sigma^2}\|x-t\|^2\right)\right)h(t)^\top dt \\
&= \frac{1}{(2\pi\sigma^2)^{d/2}}\int_{\mathbb{R}^d}\frac{t-x}{\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\|x-t\|^2\right)h(t)^\top dt \ ,
\end{aligned}$$

with a change of variable and definition of expectation, we get the result.

Lemma B.3.2.
For $L \geq 0, r \geq 0$, let $h : \mathbb{R}^d \mapsto [0, r]$ be defined as follows:

$$h(x) = \frac{1}{2}\,sign(x_1)\min\{r, 2L|x_1|\} + \frac{r}{2} \ ,$$

where sign is the sign function with the convention $sign(0) = 0$. Then, h is L-Lipschitz continuous.

Proof. To show that h is L-Lipschitz continuous, we need to demonstrate that for all $x, y \in \mathbb{R}^d$:

$$|h(x) - h(y)| \leq L\|x-y\|_2 \ .$$

We write $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$. In the following cases, only the first coordinate is going to intervene. We consider three cases:

Case 1: $x_1 = 0$ and $y_1 = 0$.
In this case, $\text{sign}(x_1) = 0$, $\text{sign}(y_1) = 0$, and $h(y) = h(x) = \frac{r}{2}$ for any x.
Thus, $|h(x) - h(y)| = 0 \leq L\|x - y\|_2$.
Case 2: $x_1 \neq 0$ and $y_1 = 0$ (without loss of generality same as $x_1 = 0$ and $y_1 \neq 0$).
In this case, $h(x)$ is given by:

$$h(x) = \frac{1}{2}\text{sign}(x_1)\min\{r, 2L|x_1|\} + \frac{r}{2} ,$$

and $h(y)$ is given by:

$$h(y) = \frac{r}{2} .$$

Now, let's consider the difference $|h(x) - h(y)|$:

$$|h(x) - h(y)| = \left|\frac{1}{2}\text{sign}(x_1)\min\{r, 2L|x_1|\} + \frac{r}{2} - \frac{r}{2}\right| .$$

If $2L|x_1| \leq r$, then $\min\{r, 2L|x_1|\} = 2L|x_1|$ and the expression becomes:

$$\left|\frac{1}{2}(2Lx_1) + \frac{r}{2} - \frac{r}{2}\right| = L|x_1| \leq L\|x - y\|_2 .$$

If $2L|x_1| > r$, then $\min\{r, 2L|x_1|\} = r$ and the expression becomes:

$$\left|\frac{1}{2}\text{sign}(x_1)r + \frac{r}{2} - \frac{r}{2}\right| = \frac{r}{2} \leq L|x_1| \leq L\|x - y\|_2 .$$

In both cases, $|h(x) - h(y)| \leq L\|x - y\|_2$, therefore, in the case where $x_1 \neq 0$ and $y_1 = 0$, $|h(x) - h(y)|$ is L-Lipschitz. Same result goes for $x_1 = 0$ and $y_1 \neq 0$.
Case 3: $x_1 \neq 0$ and $y_1 \neq 0$.
Without loss of generality, assume $|x_1| \geq |y_1|$. Consider

$$|h(x) - h(y)| = \frac{1}{2}\left|\text{sign}(x_1)\min\{r, 2L|x_1|\} - \text{sign}(y_1)\min\{r, 2L|y_1|\}\right| .$$

Let's consider two sub-cases:
Sub-case 3a: If $2L|x_1| \leq r$, then $\min\{r, 2L|x_1|\} = 2L|x_1|$.
Sub-case 3b: If $2L|x_1| > r$, then $\min\{r, 2L|x_1|\} = r$.
Similarly, for $|y_1|$, we have $\min\{r, 2L|y_1|\} = 2L|y_1|$ if $2L|y_1| \leq r$ and $\min\{r, 2L|y_1|\} = r$ if $2L|y_1| > r$.
In both sub-cases, we can write:

$$|h(x) - h(y)| = \frac{1}{2}|2Lx_1 - 2Ly_1| = L|x_1 - y_1| \leq L\|x - y\|_2 .$$

Therefore, h is L-Lipschitz continuous.

Now we are ready to prove the theorem.

Proof. For ease of notation we note $L = L(f)$, we are interested in the following:

$$J(\sigma, L) = \sup_{h:L(h)=L} \sup_{x\in\mathbb{R}^d} \|\nabla\tilde{h}(x)\|_2 = \sup_{h:L(h)=L} \sup_{x\in\mathbb{R}^d} \sup_{v\in\mathbb{R}^d:\|v\|=1} v^\top\nabla\tilde{h}(x) \ .$$

First, we will derive an upper bound on $J(\sigma, L)$. Consider any $x \in \mathbb{R}^d$, any h Lipschitz continuous s.t $h(x) \in [0, r]$, and any $v \in \mathbb{R}^d$ with $\|v\| = 1$. Any $\delta \in \mathbb{R}^d$ can be decomposed as $\delta = \delta^\perp + \tilde{\delta}$, where $\tilde{\delta} = (v^T\delta)v$ and $\delta^\perp \perp v$. Let $\delta' = \delta^\perp - \tilde{\delta}$. That is, δ' is the reflection of the vector δ with respect to the hyperplane that is normal to v. If $\delta \sim \mathcal{N}(0, \sigma^2 I)$, then $\delta' \sim \mathcal{N}(0, \sigma^2 I)$ because $\mathcal{N}(0, \sigma^2 I)$ is radially symmetric. Moreover, $v^T\delta' = -v^T\delta$. Hence,

$$\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v^\top\delta h(x+\delta)] = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v^\top\delta' h(x+\delta')] = -\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v^\top\delta h(x+\delta')] \ .$$

Using the above, we have the following, using Stein's Lemma 2.1.4:

$$\begin{aligned}
v^\top\nabla\tilde{h}(x) &= \frac{1}{\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v^T\delta h(x+\delta)] \\
&= \frac{1}{2\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v^T\delta(h(x+\delta) - h(x+\delta'))] \\
&\leq \frac{1}{2\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[|v^T\delta|\,|(h(x+\delta) - h(x+\delta'))|] \\
&\overset{(i)}{\leq} \frac{1}{2\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[|v^T\delta|\min\{r, 2L|v^T\delta|\}] \\
&\overset{(ii)}{=} \frac{1}{2\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[|\delta_1|\min\{r, 2L|\delta_1|\}] \\
&\overset{(iii)}{=} \frac{1}{2\sigma^2}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[|z|\min\{r, 2L|z|\}] \ ,
\end{aligned}$$

where (i) follows from the Lipschitz assumption on h and the fact that $\|\delta - \delta'\| = \|2\tilde{\delta}\| = \|2(v^T\delta)v\| = 2|v^T\delta|$ and that $|h(x+\delta) - h(x+\delta')| \leq r$, (ii) follows by choosing the canonical unit vector $v = (1, 0, \ldots, 0)$ because the previous expression does not depend on the direction of v, and (iii) follows by simply rewriting the expression in terms of z.

Now, we will derive a lower bound on $J(\sigma, L)$. For this, we choose a specific $\bar{h} \in [0, r]$ as $\bar{h}(x) = \frac{1}{2}\text{sign}(x_1)\min\{r, 2L|x_1|\} + r/2$, with $\text{sign}(0) = 0$. Using Lemma B.3.2, we have that $\bar{h}$ is L-Lipschitz. We choose a specific $x = 0$ and specific unit vector $v_0 = (1, 0, \ldots, 0)$. For this choice, note that $\bar{h}(0) = r/2$ and $v_0^\top\delta = \delta_1$. Then,

$$\begin{aligned}
J(\sigma, L) \geq v_0^\top\nabla\tilde{\bar{h}} &= \frac{1}{\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[v_0^\top\delta\bar{h}(\delta)^\top] \\
&= \frac{1}{2\sigma^2}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[|\delta_1|\min\{r, 2L|\delta_1|\}] + 0 \\
&= \frac{1}{2\sigma^2}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[|z|\min\{r, 2L|z|\}] \ .
\end{aligned}$$

Combining the upper and lower bounds, we have the following equality:

$$J(\sigma, L) = \frac{1}{2\sigma^2}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\min\left\{r|z|, 2Lz^2\right\}\right] . \tag{B.1}$$

We will now compute the above expression exactly:

$$\begin{aligned}
&\frac{1}{2\sigma^2}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\min\left\{r|z|, 2Lz^2\right\}\right] \\
&\qquad = \frac{1}{2\sigma^2}\int_{-\frac{r}{2L}}^{\frac{r}{2L}} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-z^2}{2\sigma^2}\right)2Lz^2dz \\
&\qquad - \frac{1}{2\sigma^2}\int_{-\infty}^{-\frac{r}{2L}} \frac{r}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-z^2}{2\sigma^2}\right)zdz \\
&\qquad + \frac{1}{2\sigma^2}\int_{\frac{r}{2L}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-z^2}{2\sigma^2}\right)zdz \\
&\qquad = \frac{1}{\sigma^2}\int_{-\frac{r}{2L}}^{\frac{r}{2L}} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-z^2}{2\sigma^2}\right)Lz^2dz + \frac{e^{-\frac{r}{8L^2\sigma^2}}}{2^{\frac{3}{2}}\sqrt{\pi}|\sigma|} + \frac{e^{-\frac{r}{8L^2\sigma^2}}}{2^{\frac{3}{2}}\sqrt{\pi}|\sigma|} \\
&\qquad = L\,\mathrm{erf}\left(\frac{r}{2^{\frac{3}{2}}L\sigma}\right) - \frac{e^{-\frac{r}{8L^2\sigma^2}}}{\sqrt{2}\sqrt{\pi}|\sigma|} + \frac{e^{-\frac{r}{8L^2\sigma^2}}}{2^{\frac{3}{2}}\sqrt{\pi}|\sigma|} + \frac{e^{-\frac{r}{8L^2\sigma^2}}}{2^{\frac{3}{2}}\sqrt{\pi}|\sigma|} \\
&\qquad = L\,\mathrm{erf}\left(\frac{r}{2^{\frac{3}{2}}L\sigma}\right) .
\end{aligned}$$

:*Jensen's inequality gives the following simple upper bound on* $J(\sigma, L)$*:*

$$\begin{aligned}
J(\sigma, L) &= \frac{1}{2\sigma^2}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\min\left\{r|z|, 2Lz^2\right\}\right] \\
&\leq \min\left\{\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[r|z|/2\sigma^2], \mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[Lz^2/\sigma^2]\right\} \\
&\leq \min\left\{\frac{r}{\sqrt{2\pi\sigma^2}}, L\right\} .
\end{aligned}$$

Hence, $J(\sigma, L)$ *is no worse than the Lipschitz constant of the original classifier* f*, or its Gaussian smoothed counterpart without the Lipschitz assumption, the latter bound is twice smaller than the previous original derivation in Salman et al., 2019, Appendix A.*

B.4 Proof of Proposition 4.6.3 optimal smoothing parameter for Lipschitz continuity

Proof. For $\gamma \geq 0$, we seek σ^* that maximizes the gap between the bounds of Eq. (4.2) with respect to σ:

$$\sigma^* = \arg\max_{\sigma>0} \left\{ \min \left\{ \gamma, \frac{r}{\sqrt{2\pi\sigma^2}} \right\} - \gamma \operatorname{erf}\left(\frac{r}{2^{3/2}\gamma\sigma} \right) \right\} .$$

To find the value of σ^* that maximizes the given function, we'll determine the critical points. Let

$$g(\sigma) = \min \left\{ \gamma, \frac{r}{\sqrt{2\pi\sigma^2}} \right\} - \gamma \operatorname{erf}\left(\frac{r}{2^{3/2}\gamma\sigma} \right) .$$

Let's start by setting the two functions inside the $\min$ function equal to each other and solving for σ:

$$\begin{aligned} \gamma &= \frac{r}{\sqrt{2\pi\sigma^2}} \\ \Rightarrow \sigma^2 &= \frac{r}{\gamma^2 2\pi} \\ \Rightarrow \sigma &= \frac{r}{\gamma\sqrt{2\pi}} . \end{aligned}$$

This is the point of intersection, hence the value of σ where the two functions inside the $\min$ change dominance. For that value, $g(\frac{r}{\gamma\sqrt{2\pi}}) = \gamma(1 - \operatorname{erf}(\frac{\sqrt{\pi}}{2}))$.

Now, for $\sigma < \frac{r}{\gamma\sqrt{2\pi}}$, $g(\sigma) = \gamma - \gamma \operatorname{erf}\left(\frac{r}{2^{3/2}\gamma\sigma}\right)$.

Let's differentiate $g(\sigma)$ in the this first region:

$$\begin{aligned} g'(\sigma) &= 0 - \frac{d}{d\sigma}\left[\gamma \operatorname{erf}\left(\frac{r}{2^{3/2}\gamma\sigma} \right) \right] \\ &= \gamma \frac{r}{2^{3/2}\gamma} \frac{2}{\sqrt{\pi}} \exp\left(-\left(\frac{r}{2^{3/2}\gamma\sigma} \right)^2 \right) \\ &= \frac{r}{\sqrt{2\pi}} \exp\left(-\left(\frac{r}{2^{3/2}\gamma\sigma} \right)^2 \right) . \end{aligned}$$

The supremum is obtained for $\sigma \to 0$, and the associated limit value for $g(\sigma)$ is 0.

For the second region, $\sigma > \frac{r}{\gamma\sqrt{2\pi}}$, $g(\sigma) = \frac{r}{\sqrt{2\pi\sigma^2}} - \gamma \operatorname{erf}\left(\frac{r}{2^{3/2}\gamma\sigma}\right)$. Supremum is obtained for $\sigma \to \frac{r}{\gamma\sqrt{2\pi}}$ as g is a decreasing function of σ. The associated limit value for $g(\sigma)$ is $\gamma(1 - \operatorname{erf}(\frac{\sqrt{\pi}}{2}))$.

Finally, taking $\gamma = L(s_k^r \circ f)$, $\sigma^* = \frac{r}{L(s_k^r \circ f)\sqrt{2\pi}}$ gives maximum value for g on all domain.

We get a similar result $\sigma^* = \frac{r}{L(s^r \circ f)\sqrt{\pi}}$ for $L(s^r \circ f)$.

B.5 Proof of Theorem 4.6.5 Weierstrass transform with local Lipschitz continuity

We recall the Pontryagin's maximum principle (PMP).

Theorem B.5.1 (Pontryagin's Maximum Principle)**.**
(Pontryagin et al., 1962) Consider the optimal control problem:

Minimize (or maximize) the cost functional

$$J(u) = \int_{t_0}^{t_f} L(t, x(t), u(t))\, dt + \Phi(x(t_f)),$$

subject to the state (dynamical) equations

$$\dot{x}(t) = f(t, x(t), u(t)), \quad x(t_0) = x_0,$$

and the control constraints

$$u(t) \in U \subset \mathbb{R}^m,$$

where $x(t) \in \mathbb{R}^n$*,* $u(t) \in \mathbb{R}^m$*, and* $t \in [t_0, t_f]$*.*

Assume that L*,* f*, and* Φ *are continuously differentiable with respect to their arguments and that an optimal control* $u^*(t)$ *and corresponding state trajectory* $x^*(t)$ *exist.*

Then there exists an absolutely continuous costate (adjoint) function $p(t) \in \mathbb{R}^n$ *and a constant* $p_0 \leq 0$ *(for minimization) or* $p_0 \geq 0$ *(for maximization) such that the following conditions are satisfied almost everywhere on* $[t_0, t_f]$*:*

1. State equation:

$$\dot{x}^*(t) = f\big(t, x^*(t), u^*(t)\big).$$

2. Costate equation:

$$\dot{p}(t) = -\frac{\partial H}{\partial x}\big(t, x^*(t), u^*(t), p(t)\big),$$

with terminal condition

$$p(t_f) = \frac{\partial \Phi}{\partial x}\big(x^*(t_f)\big).$$

The Hamiltonian H *is defined by*

$$H(t, x, u, p) = L(t, x, u) + p^\top f(t, x, u).$$

Maximum principle (optimality condition):

$$H\big(t, x^*(t), u^*(t), p(t)\big) = \max_{v \in U} H\big(t, x^*(t), v, p(t)\big).$$

Transversality conditions: If the final time t_f is free, then

$$H\big(t_f, x^*(t_f), u^*(t_f), p(t_f)\big) = 0.$$

Non-triviality condition:

$$(p_0, p(t)) \neq 0 \quad \textit{for all} \quad t \in [t_0, t_f].$$

We are going to prove the following lemma

Lemma B.5.2.
Given the optimal control problem:

Maximize the functional

$$I(y) = \int_{-\infty}^{+\infty} sy(s)\phi(s)\, ds$$

subject to the state equation

$$\frac{dy(s)}{ds} = u(s),$$

the control constraint

$$|u(s)| \leq L,$$

the state constraints

$$0 \leq y(s) \leq 1,$$

and the integral constraint

$$\int_{-\infty}^{+\infty} y(s)\phi(s)\, ds = p,$$

where $\phi(s)$ is the standard normal probability density function (PDF), $\Phi(s)$ is the cumulative distribution function (CDF) and $p \in [0, 1]$.

The optimal control $u^*(s)$ and the optimal state $y^*(s)$ are given by:

When $s \leq s_0$:

$$y^*(s) = 0, \quad u^*(s) = 0.$$

When $s_0 < s < s_1$*:*

$$y^*(s) = L(s - s_0), \quad u^*(s) = L.$$

When $s \geq s_1$*:*

$$y^*(s) = 1, \quad u^*(s) = 0.$$

Here, s_0 *is determined by the equation*

$$p - \left(1 - L \int_{s_0}^{s_0 + \frac{1}{L}} \Phi(s)\, ds\right) = 0.$$

The optimal value of the objective functional $I(y^*)$ *is:*

$$I(y^*) = L[\Phi(s_0 + \frac{1}{L}) - \Phi(s_0)].$$

Proof. To solve the optimal control problem, we apply the PMP with state constraints incorporated via Lagrange multipliers. To formulate the Hamiltonian with state constraints, we introduce Lagrange multipliers $\mu_0(s) \geq 0$ and $\mu_1(s) \geq 0$ for the state constraints $y(s) \geq 0$ and $y(s) \leq 1$, respectively.
The augmented Hamiltonian is:

$$H(s, y, p_y, u, \mu_0, \mu_1) = [sy - \lambda y]\phi(s) + p_y u + \mu_0 y + \mu_1(-y + 1),$$

where λ is the Lagrange multiplier associated with the integral constraint.
Necessary Conditions from the PMP are:
State equation:

$$\frac{dy(s)}{ds} = u(s).$$

Costate equation:

$$\frac{dp_y(s)}{ds} = -\frac{\partial H}{\partial y} = -[s - \lambda]\phi(s) - \mu_0(s) + \mu_1(s).$$

Hamiltonian maximization condition:

$$u^*(s) = \arg\max_{|u(s)| \leq L} [p_y(s)u(s)].$$

Complementary slackness conditions:

$$\mu_0(s) \geq 0, \quad y(s) \geq 0, \quad \mu_0(s)y(s) = 0,$$
$$\mu_1(s) \geq 0, \quad y(s) \leq 1, \quad \mu_1(s)[y(s) - 1] = 0.$$

We consider three regions based on the value of $y(s)$:

Region where $y(s) = 0$ (lower state constraint active): Complementary slackness: $\mu_0(s) \geq 0$, $\mu_1(s) = 0$.

Costate equation:

$$\frac{dp_y(s)}{ds} = -[s - \lambda]\phi(s) - \mu_0(s).$$

Admissible controls: $u(s) \geq 0$ (to prevent $y(s)$ from decreasing below zero).

Optimal control: Since $p_y(s) \leq 0$ (due to the effect of $\mu_0(s)$), the Hamiltonian is maximized by $u^*(s) = 0$.

Resulting State: $y(s) = 0$.

Region where $0 < y(s) < 1$ (state constraints inactive): Complementary slackness: $\mu_0(s) = 0$, $\mu_1(s) = 0$.

Costate equation:

$$\frac{dp_y(s)}{ds} = -[s - \lambda]\phi(s).$$

Solution of costate equation:

$$p_y(s) = -\int_s^{+\infty} [\sigma - \lambda]\phi(\sigma)\, d\sigma = \lambda Q(s) - \phi(s),$$

where $Q(s) = \int_s^{+\infty} \phi(\sigma)\, d\sigma$.

The optimal control is:

$$u^*(s) = \begin{cases} L, & \text{if } p_y(s) > 0, \\ -L, & \text{if } p_y(s) < 0. \end{cases}$$

We determining the switching point s_0:

$$p_y(s_0) = 0 \implies \lambda Q(s_0) = \phi(s_0).$$

Since $p_y(s) > 0$ for $s > s_0$, the optimal control in this region is $u^*(s) = L$.

The resulting state is:

$$y(s) = y(s_0) + L(s - s_0).$$

Region where $y(s) = 1$ (upper state constraint active): Complementary slackness: $\mu_0(s) = 0$, $\mu_1(s) \geq 0$.

Costate equation:

$$\frac{dp_y(s)}{ds} = -[s - \lambda]\phi(s) + \mu_1(s).$$

Admissible controls: $u(s) \leq 0$ (to prevent $y(s)$ from increasing above one).

Optimal control: since $p_y(s) \geq 0$ (due to the effect of $\mu_1(s)$), the Hamiltonian is maximized by $u^*(s) = 0$.

Resulting state: $y(s) = 1$.

Determining s_0 and s_1:

At $s = s_0$, $y(s_0) = 0$ (transition from $y(s) = 0$ to $y(s) > 0$). At $y(s_1) = 1$, $s_1 = s_0 + \frac{1}{L}$.

Integral constraint:

$$p = \int_{s_0}^{s_1} y(s)\phi(s)\,ds + \int_{s_1}^{+\infty} \phi(s)\,ds.$$

Solving for λ, s_0, and s_1 requires numerical methods due to the integrals involved. By incorporating the state constraints into the Hamiltonian via Lagrange multipliers and applying the necessary conditions from the PMP, we derive the optimal control and state trajectories that satisfy all constraints:

Optimal Control $u^*(s)$:

$$u^*(s) = \begin{cases} 0, & s \le s_0, \\ L, & s_0 < s < s_1, \\ 0, & s \ge s_1. \end{cases}$$

Optimal state $y^*(s)$:

$$y^*(s) = \begin{cases} 0, & s \le s_0, \\ L(s - s_0), & s_0 < s < s_1, \\ 1, & s \ge s_1. \end{cases}$$

This solution ensures that the Hamiltonian is maximized over the admissible controls while satisfying the state and control constraints.

We will derive the expressions for p and $I(y^*)$ in terms of s_0, L, and the standard normal CDF $\Phi(s)$.

Expression for p: The integral constraint is:

$$p = \int_{-\infty}^{+\infty} y^*(s)\phi(s)\,ds.$$

Using the structure of $y^*(s)$:

$$y^*(s) = \begin{cases} 0, & s \le s_0, \\ L(s - s_0), & s_0 < s < s_1, \\ 1, & s \ge s_1. \end{cases}$$

We can split the integral into regions:

$$p = \int_{s_0}^{s_1} y^*(s)\phi(s)\,ds + \int_{s_1}^{+\infty} y^*(s)\phi(s)\,ds.$$

Compute each part:

first integral (p_1):

$$p_1 = \int_{s_0}^{s_1} L(s - s_0)\phi(s)\,ds = L\int_{s_0}^{s_1} (s - s_0)\phi(s)\,ds.$$

We perform integration by parts:

Let $u(s) = (s - s_0)$, $dv(s) = \phi(s)\,ds$.
Then $du(s) = ds$, $v(s) = \Phi(s)$.
Integration by parts gives:

$$\int_{s_0}^{s_1} (s - s_0)\phi(s)\,ds = \left[(s - s_0)\Phi(s)\right]_{s_0}^{s_1} - \int_{s_0}^{s_1} \Phi(s)\,ds.$$

Evaluating the boundaries:

$$(s_1 - s_0)\Phi(s_1) - (s_0 - s_0)\Phi(s_0) = \frac{1}{L}\Phi(s_1).$$

Therefore,

$$\int_{s_0}^{s_1} (s - s_0)\phi(s)\,ds = \frac{1}{L}\Phi(s_1) - \int_{s_0}^{s_1} \Phi(s)\,ds.$$

Multiply both sides by L:

$$p_1 = \Phi(s_1) - L\int_{s_0}^{s_1} \Phi(s)\,ds.$$

Second integral (p_2):

$$p_2 = \int_{s_1}^{+\infty} 1 \cdot \phi(s)\,ds = 1 - \Phi(s_1).$$

Total p:

$$p = p_1 + p_2 = \Phi(s_1) - L\int_{s_0}^{s_1} \Phi(s)\,ds + 1 - \Phi(s_1) = 1 - L\int_{s_0}^{s_1} \Phi(s)\,ds.$$

Expression for $I(y^*)$: The objective functional evaluated at $y^*(s)$ is:

$$I(y^*) = \int_{-\infty}^{+\infty} s y^*(s)\phi(s)\,ds.$$

Again, split the integral:

$$I(y^*) = \int_{s_0}^{s_1} s y^*(s)\phi(s)\,ds + \int_{s_1}^{+\infty} s y^*(s)\phi(s)\,ds.$$

Compute each part:
First integral (I_1):

$$I_1 = \int_{s_0}^{s_1} s[L(s - s_0)]\phi(s)\,ds = L\int_{s_0}^{s_1} s(s - s_0)\phi(s)\,ds.$$

Expand $s(s - s_0) = s^2 - s_0 s$:

$$I_1 = L\left(\int_{s_0}^{s_1} s^2\phi(s)\,ds - s_0\int_{s_0}^{s_1} s\phi(s)\,ds\right).$$

Recall standard integrals:

$$\int s^2\phi(s)\,ds = -s\phi(s) + \Phi(s),$$
$$\int s\phi(s)\,ds = -\phi(s).$$

Evaluate the integrals:

$$\int_{s_0}^{s_1} s^2\phi(s)\,ds = [-s\phi(s) + \Phi(s)]_{s_0}^{s_1} = [-s_1\phi(s_1) + \Phi(s_1)] - [-s_0\phi(s_0) + \Phi(s_0)],$$
$$\int_{s_0}^{s_1} s\phi(s)\,ds = [-\phi(s)]_{s_0}^{s_1} = -\phi(s_1) + \phi(s_0).$$

Substitute back into I_1:

$$\begin{aligned} I_1 &= L\left([-s_1\phi(s_1) + \Phi(s_1)] - [-s_0\phi(s_0) + \Phi(s_0)] - s_0(-\phi(s_1) + \phi(s_0))\right) \\ &= L\left(\Phi(s_1) - \Phi(s_0) - s_1\phi(s_1) + s_0\phi(s_0) + s_0\phi(s_1) - s_0\phi(s_0)\right) \\ &= L\left(\Phi(s_1) - \Phi(s_0) - (s_1 - s_0)\phi(s_1)\right). \end{aligned}$$

Since $s_1 - s_0 = \frac{1}{L}$:

$$I_1 = L\left(\Phi(s_1) - \Phi(s_0) - \frac{1}{L}\phi(s_1)\right) = L[\Phi(s_1) - \Phi(s_0)] - \phi(s_1).$$

Second integral (I_2):

$$I_2 = \int_{s_1}^{+\infty} s \cdot 1 \cdot \phi(s)\,ds.$$

Recall that:

$$\int_{s}^{+\infty} s\phi(s)\,ds = \phi(s).$$

Therefore:

$$I_2 = \phi(s_1).$$

Total $I(y^*)$:

$$I(y^*) = I_1 + I_2 = (L[\Phi(s_1) - \Phi(s_0)] - \phi(s_1)) + \phi(s_1) = L[\Phi(s_1) - \Phi(s_0)].$$

Thus, the optimal value of the objective functional is:

$$I(y^*) = L[\Phi(s_1) - \Phi(s_0)].$$

Now we can prove Theorem 2.

Proof. Let us assume $\sigma = 1$. We start by expressing the gradient of $\phi^{-1}(\tilde{F}(\mathbf{x}))$:

$$\nabla\phi^{-1}(\tilde{F}(\mathbf{x})) = \frac{\nabla\tilde{F}(\mathbf{x})}{\phi'(\phi^{-1}(\tilde{F}(\mathbf{x})))}\,.$$

We aim to show that for any unit vector $\mathbf{u}$, the following inequality holds:

$$\mathbf{u}^\top \nabla \tilde{F}(\mathbf{x}) \leq L(F) \left[\Phi\left(s_0 + \frac{1}{L(F)} \right) - \Phi(s_0) \right],$$

where s_0 is determined by:

$$\tilde{F}(\mathbf{x}) = 1 - L(F) \int_{s_0}^{s_0 + \frac{1}{L(F)}} \Phi(s)\, ds\ .$$

Applying Stein's Lemma

Using Stein's lemma, we obtain the expression for $\mathbf{u}^\top \nabla \tilde{F}(\mathbf{x})$:

$$\mathbf{u}^\top \nabla \tilde{F}(\mathbf{x}) = \mathbb{E}_{\delta \sim \mathcal{N}(0,\mathbf{I})} \left[\mathbf{u}^\top \delta F(\mathbf{x} + \delta) \right],$$

where δ is a Gaussian vector with mean zero and identity covariance matrix. Our goal is to bound the maximum of this expression under the following constraints:

- $0 \leq F(\mathbf{x}) \leq 1$
- $\mathbb{E}_{\delta \sim \mathcal{N}(0,\mathbf{I})} \left[F(\mathbf{x} + \delta) \right] = p$
- F is Lipschitz continuous with a Lipschitz constant $L(F)$.

Let $y(z) = F(z + \mathbf{x})$. Then the problem can be recast as:

$$(P) \quad \max_{\|\mathbf{u}\|=1} \max_{y} \quad I(y) = \int_{\mathbb{R}^d} \mathbf{u}^\top s\, y(s)\, \phi(s)\, ds$$

subject to:

$$L(y) \leq L(F), \quad 0 \leq y(s) \leq 1, \quad \int_{\mathbb{R}^d} y(s)\phi(s)\, ds = p.$$

Without loss of generality, we can take $\mathbf{u} = (1, 0, \ldots, 0)^\top$, as Gaussian vectors are rotationally invariant and rotation preserves the ℓ^2-norm.

Reducing to a One-Dimensional Problem

Decompose $\mathbf{s}$ as $\mathbf{s} = (s_1, \mathbf{t})$, where $s_1 = \mathbf{u}^\top \mathbf{s}$ and $\mathbf{t} = (\mathbf{s}_2, \ldots, \mathbf{s}_d)$. The density function $\phi(\mathbf{s})$ can be factorized as:

$$\phi(\mathbf{s}) = \phi_1(s_1)\phi_{d-1}(\mathbf{t}),$$

where ϕ_1 and ϕ_{d-1} are the density functions of a standard Gaussian in one dimension and $d-1$ dimensions, respectively.

Rewrite the objective function as:

$$I(y) = \int_{\mathbb{R}^{d-1}} \int_{\mathbb{R}} s_1 y(s_1, \mathbf{t}) \phi_1(s_1) \phi_{d-1}(\mathbf{t})\, ds_1\, d\mathbf{t}.$$

Define $g(s_1) = \int_{\mathbb{R}^{d-1}} y(s_1, \mathbf{t}) \phi_{d-1}(\mathbf{t})\, d\mathbf{t}$. Then, the integral becomes:

$$I(y) = I(g) = \int_{\mathbb{R}} s_1 g(s_1) \phi_1(s_1)\, ds_1.$$

Reformulating the Constraints for $g(s_1)$

The integral constraint on p directly translates to:

$$\int_{\mathbb{R}} g(s_1)\phi_1(s_1)\,ds_1 = p.$$

Since $0 \le y(\mathbf{s}) \le 1$ for all $\mathbf{s} \in \mathbb{R}^d$, it follows that $0 \le g(s_1) \le 1$ for all $s_1 \in \mathbb{R}$.

For the Lipschitz condition, we have:

$$|\nabla g(s_1)| \le \int_{\mathbb{R}^{d-1}} |\nabla y(s_1, \mathbf{t})|\phi_{d-1}(\mathbf{t})\,d\mathbf{t} \le L(F).$$

Solving the One-Dimensional Problem

The reduced problem is now:

$$(Q) \quad \max_g \quad I(g) = \int_{\mathbb{R}} s_1 g(s_1)\phi_1(s_1)\,ds_1$$

subject to:

$$L(g) \le L(F), \quad 0 \le g(s_1) \le 1, \quad \int_{\mathbb{R}} g(s_1)\phi_1(s_1)\,ds_1 = p.$$

Applying Lemma B.5.2, the optimal solution $g^*(s_1)$ is given by:

$$g^*(s_1) = \begin{cases} 0, & s_1 \le s_0, \\ L(F)(s_1 - s_0), & s_0 < s_1 < s_0 + \frac{1}{L(F)}, \\ 1, & s_1 \ge s_0 + \frac{1}{L(F)}. \end{cases}$$

Here, s_0 is determined by the equation:

$$\tilde{F}(\mathbf{x}) = 1 - L(F) \int_{s_0}^{s_0 + \frac{1}{L(F)}} \Phi(s)\,ds.$$

Evaluating $I(g^*)$, we find:

$$\mathbf{u}^\top \nabla \tilde{F}(\mathbf{x}) \le L(F)\left[\Phi\left(s_0 + \frac{1}{L(F)}\right) - \Phi(s_0)\right].$$

For arbitrary σ, the result follows by scaling the variables appropriately and performing a change of integration variable.

Finally,

$$L\left(\Phi^{-1} \circ \tilde{F}, B(\mathbf{x}, \rho)\right) = \sup_{\mathbf{x}' \in B(\mathbf{x},\rho)} \left\{ \frac{L(F)\left[\Phi_\sigma\left(s_0(\mathbf{x}') + \frac{1}{L(F)}\right) - \Phi_\sigma(s_0(\mathbf{x}'))\right]}{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\Phi^{-1}(\tilde{F}(\mathbf{x}'))\right)^2\right)} \right\}$$

C

Risk management

C.1 Counterexample for Bonferroni correction

The Bonferroni correction is a method used to adjust the significance level of statistical tests when multiple comparisons are made simultaneously. It helps control the family-wise error rate by reducing the risk of false positives due to multiple hypothesis testing. This section provides a counterexample to illustrate the importance of properly applying the Bonferroni correction in statistical inference. The corresponding code can be found in the following repository: counter example. The code defines a function to compute the probability of an event under a multinomial distribution and demonstrates the effect of applying and not applying the Bonferroni correction.

Proper Bonferroni Correction: Adjusts the significance level α by dividing it by the number of comparisons (e.g., $\alpha' = \alpha/k$) to control the family-wise error rate.

Improper Application: Using the unadjusted α fails to account for multiple comparisons, leading to narrower confidence intervals and an increased risk of Type I errors.

When the Bonferroni correction is properly applied, the calculated probability of the event meets or exceeds the theoretical minimum probability $(1 - \alpha)$, maintaining the desired confidence level. Without the correction, the probability of the event may fall below the theoretical minimum, indicating that the confidence intervals are too narrow and do not provide the intended level of confidence. Properly adjusting for multiple comparisons is crucial for valid statistical inference, especially when making simultaneous inferences about multiple parameters. Ensuring accurate confidence intervals helps maintain the integrity of research findings and prevents the reporting of false positives.

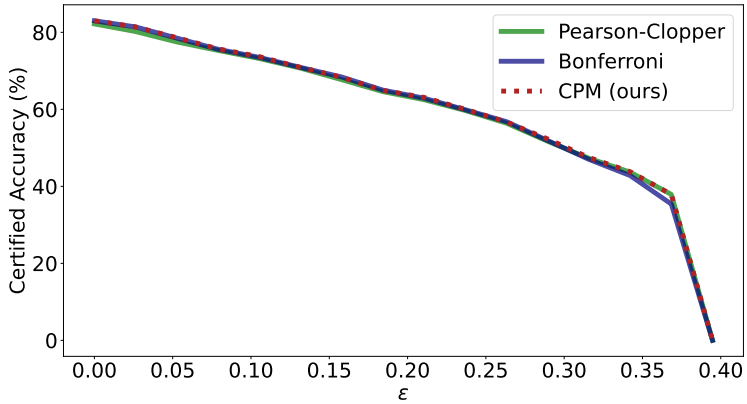


Fig. C.2.1.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 0.12$ on the CIFAR-10 dataset.

C.2 Additional experiments on CPM

We conduct additional experiments on certified prediction margin (CPM) using ResNet-110 for CIFAR-10 and ResNet-50 for ImageNet, both trained with noise injection. The training procedure follows the approach of Cohen et al. (2019), and all settings remain consistent with those described in Section 4.1 of the paper.

For CIFAR-10, we report certified accuracy for ResNet-110, trained with different noise standard deviations (0.12, 0.25, 0.5, 1), comparing confidence intervals from Pearson-Clopper, Bonferroni, and CPM. The results are illustrated in Figure C.2.1, Figure C.2.2, Figure 6.4, and Figure C.2.3.

For ImageNet, we evaluate certified accuracy for ResNet-50, trained with different noise standard deviations(0.25, 1), using the same confidence intervals. The results are shown in Figure C.2.4,and Figure C.2.5. The figure for $\sigma = 0.5$ is already presented in the main body.

The results indicate that for high variance in probability outputs, often associated with larger noise and higher entropy, Bonferroni performs better than Pearson-Clopper, while CPM achieves results comparable to Bonferroni. Conversely, when the probability outputs exhibit lower entropy, typically with smaller noise levels, Pearson-Clopper performs better, and CPM closely mimics Pearson-Clopper while also providing superior results in intermediate scenarios.

We observed slightly better gains on ImageNet, likely due to the larger number of classes, where CPM has a more significant impact, as Bonferroni tends to be overly conservative in such settings.

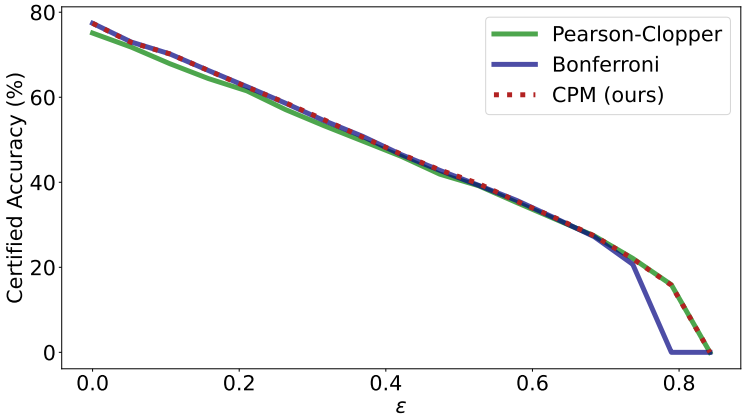


Fig. C.2.2.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 0.25$ on the CIFAR-10 dataset.

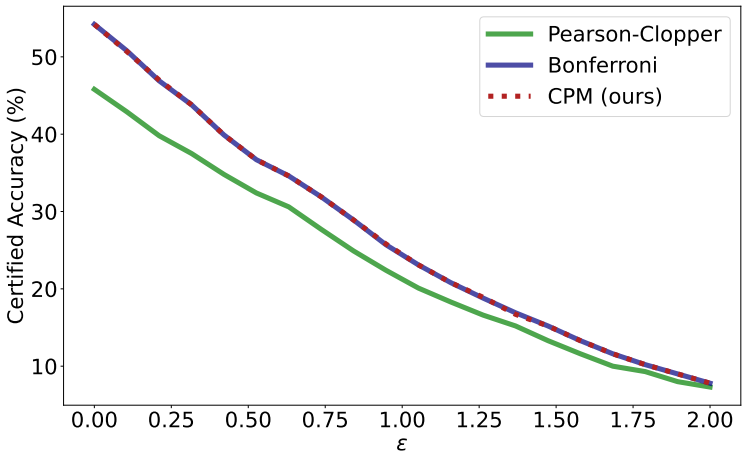


Fig. C.2.3.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 1.0$ on the CIFAR-10 dataset.

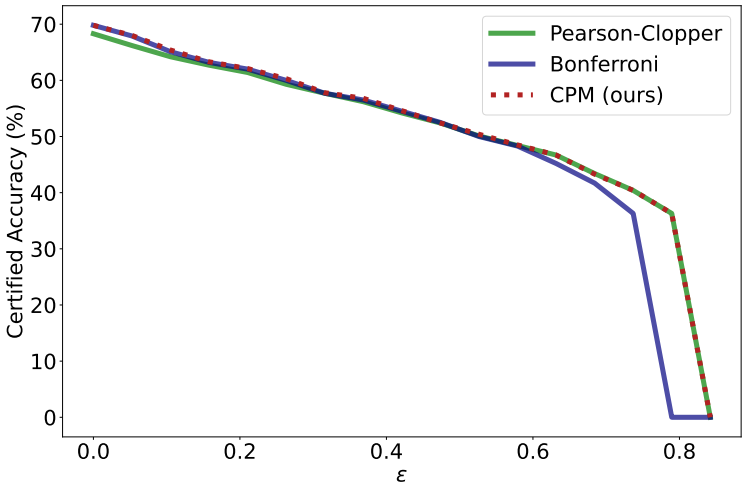


Fig. C.2.4.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 0.25$ on the ImageNet dataset.

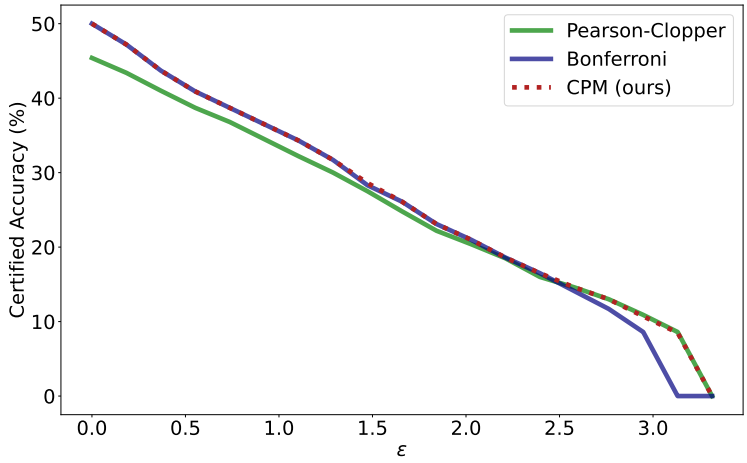


Fig. C.2.5.: Comparison of various confidence interval methods for certified accuracy estimation with smoothing standard deviation $\sigma = 1.0$ on the ImageNet dataset.

C.3 Example on hardmax

Example C.3.1 (Variance inflation by hard threshold)**.**
Let X_ϵ be a random variable uniformly distributed on the interval

$$\left(\tfrac{1}{2} - \epsilon,\ \tfrac{1}{2} + \epsilon\right), \quad \textit{where } 0 < \epsilon \leq \tfrac{1}{2}.$$

Define a new random variable

$$Y_\epsilon = s(X_\epsilon) = \mathbb{1}_{\{X_\epsilon > \frac{1}{2}\}} \quad (\textit{hard threshold at } \tfrac{1}{2}).$$

Then

$$\mathbb{V}(X_\epsilon) = \frac{\epsilon^2}{3} \quad \textit{while} \quad \mathbb{V}(Y_\epsilon) = \tfrac{1}{4}.$$

Consequently, as $\epsilon \to 0$, $\mathbb{V}(X_\epsilon) \to 0$, but $\mathbb{V}(Y_\epsilon)$ remains at $\frac{1}{4}$, showing how the hard threshold can inflate a small variance into its maximal Bernoulli value.

Proof. **Step 1 (Mean and Variance of X_ϵ):** By construction, $X_\epsilon \sim \text{Uniform}\left(\frac{1}{2} - \epsilon,\ \frac{1}{2} + \epsilon\right)$. The mean of a uniform random variable on (a, b) is $\frac{a+b}{2}$, so

$$\mathbb{E}[X_\epsilon] = \frac{\left(\frac{1}{2} - \epsilon\right) + \left(\frac{1}{2} + \epsilon\right)}{2} = \frac{1}{2}.$$

Its variance is $\frac{(b-a)^2}{12}$; hence

$$\mathbb{V}(X_\epsilon) = \frac{(2\epsilon)^2}{12} = \frac{\epsilon^2}{3}.$$

Step 2 (Distribution of Y_ϵ): The hard-threshold map

$$s(x) = \begin{cases} 1, & x > 1/2, \\ 0, & x \le 1/2 \end{cases}$$

transforms X_ϵ into a Bernoulli-type variable Y_ϵ. Because X_ϵ is *uniform* on $\left(\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon\right)$, exactly half of that interval lies above $1/2$. Consequently,

$$\mathbb{P}\big(X_\epsilon > 1/2\big) = \frac{1}{2}, \quad \mathbb{P}\big(X_\epsilon \le 1/2\big) = \frac{1}{2},$$

so $Y_\epsilon \sim \mathrm{Bernoulli}\big(1/2\big)$.

Step 3 (Mean and Variance of Y_ϵ): A $\mathrm{Bernoulli}\big(1/2\big)$ random variable Y_ϵ has:

$$\mathbb{E}[Y_\epsilon] = \frac{1}{2}, \quad \mathbb{V}(Y_\epsilon) = \frac{1}{2}\left(1 - \tfrac{1}{2}\right) = \frac{1}{4}.$$

Therefore we obtain

$$\mathbb{V}(X_\epsilon) = \frac{\epsilon^2}{3} \quad \text{and} \quad \mathbb{V}(Y_\epsilon) = \frac{1}{4}.$$

As ϵ shrinks, $\mathbb{V}(X_\epsilon) \to 0$, yet $\mathbb{V}(Y_\epsilon)$ remains at $\frac{1}{4}$, which is the maximal value for a $\mathrm{Bernoulli}(1/2)$.

This example illustrates the key point: *discontinuous transformations* (like a hard threshold) can magnify small input variations into large output variations, since the output "jumps" from 0 to 1. Such a function can thus inflate the variance of the resulting variable significantly—even to its maximal Bernoulli value $1/4$ when the mean is $1/2$.

C.4 Ablation study in Lipschitz variance margin trade-off

This ablation study provides two comparisons:

- A comparison between corrected certified radii produced by Hoeffding's and Bernstein's inequalities in Fig. C.4.1. The Clopper-Pearson is not included as it is only applicable to binomial values.

- A comparison between corrected certified radii produced by different simplex maps and temperatures in Fig.C.4.2.

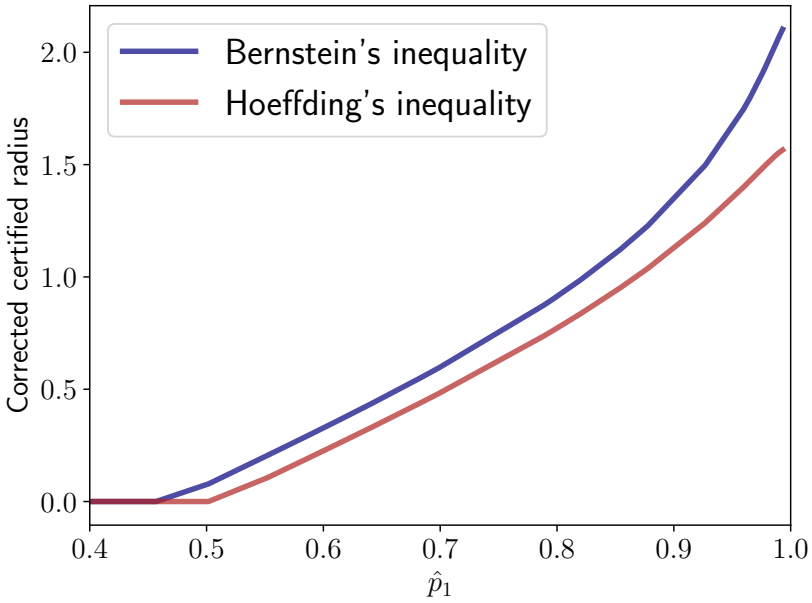


Fig. C.4.1.: Comparison between corrected certified radii $R_2(\bar{\mathbf{p}})$ produced by Bernstein's and Hoeffding's inequalities, for a random subset of 1000 images of ImageNet dataset using RS with a smoothing noise $\sigma = 1.0$. We use the ViT-denoiser baseline from Carlini et al., 2023.

C.5 LVM-RS additional experiments 6.2.5

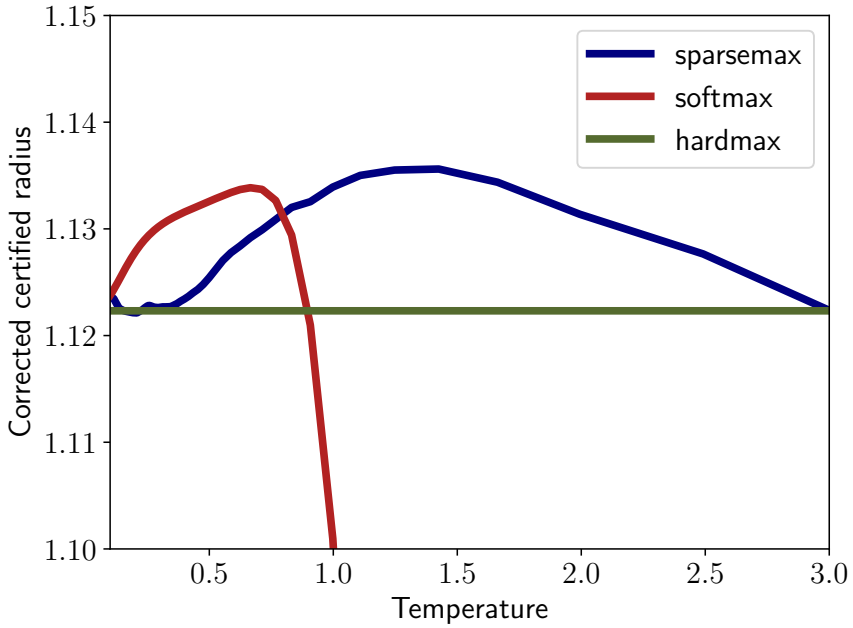


Fig. C.4.2.: Comparison of the effect on corrected certified radii $\mathrm{R}_{\mathrm{mult}}(\underline{\mathbf{p}_{I_1}}, \overline{\mathbf{p}_{I_2}})$ of the choice of the simplex map τ and associated temperature T. Simplex maps considered are $s \in \{\mathrm{sparsemax}, \mathrm{softmax}, \mathrm{hardmax}\}$. The base network f is the one from Carlini et al., 2023 and the corrected certified radii were generated with one image from ImageNet with smoothing variance $\sigma = 1.0$. Radii are risk corrected with Empirical Bernstein inequality for a risk $\alpha = 1\mathrm{e}{-3}$ and $n = 10^4$. We see that by varying the temperature T, $\mathrm{softmax}$ and $\mathrm{sparsemax}$ can find a better solution than $\mathrm{hardmax}$ to the variance-margin trade-off.

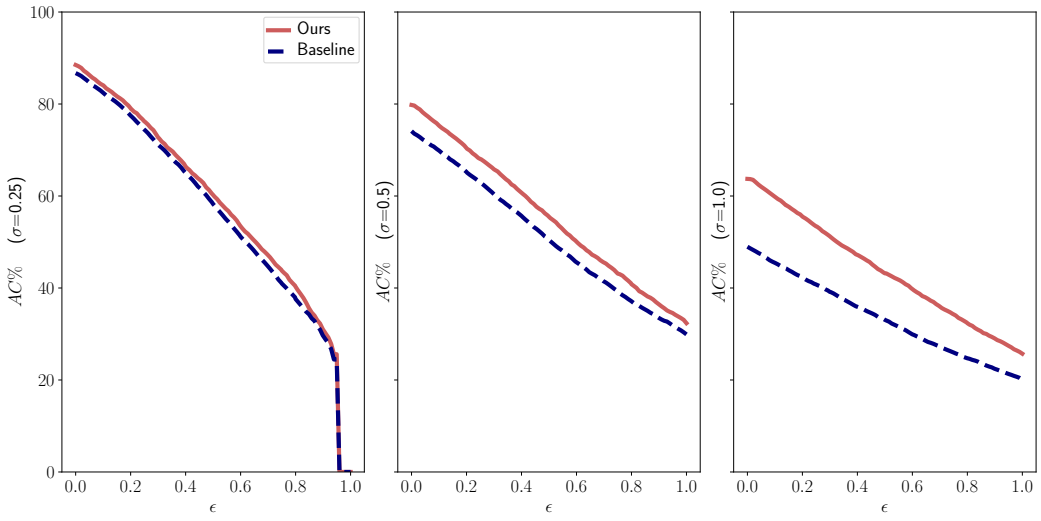


Fig. C.5.1.: Certified accuracies (CA in %) in function of level of perturbations ϵ on CIFAR-10, for different noise levels $\sigma \in \{0.25, 0.5, 1\}$. Number of samples is $n = 10^5$ and risk $\alpha = 1e\text{-}3$. Our method is compared to the baseline chosen as in Carlini et al., 2023.

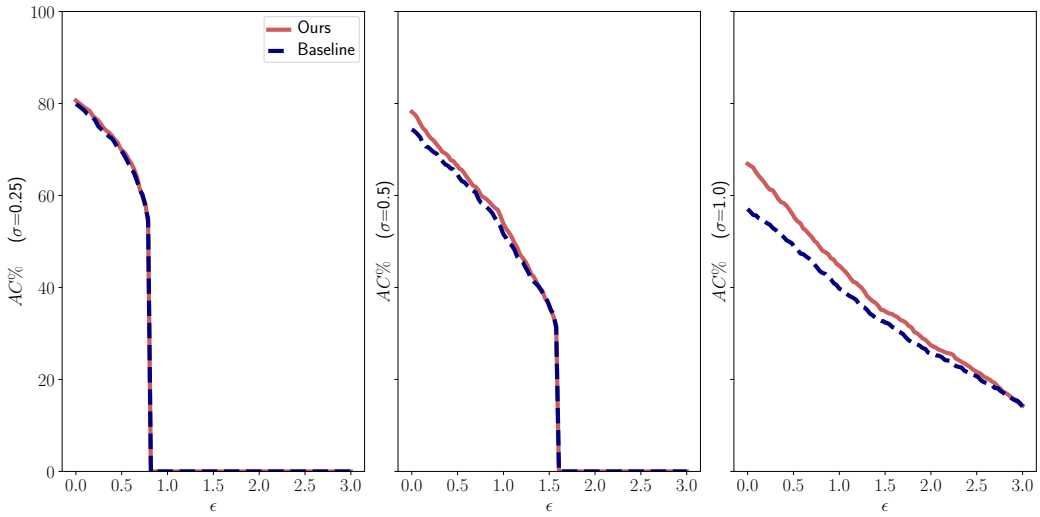


Fig. C.5.2.: Certified accuracies (CA in %) in function of level of perturbations ϵ on ImageNet, for different noise levels $\sigma \in \{0.25, 0.5, 1\}$. Number of samples is $n = 10^4$ and risk $\alpha = 1e\text{-}3$. Our method is compared to the baseline chosen as in Carlini et al., 2023.

Tab. C.5.1.: Certified accuracies comparison for different perturbation ϵ values, for $n = 10^4$ samples and $\alpha = 1\text{e}{-3}$. On ImageNet dataset. Here the baseline is a ResNet-150 from Salman et al., 2019.

Methods	**Certified accuracy (ε)**							
	0.14	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Salman et al., 2019	74.49	73.08	69.84	66.41	62.42	57.75	51.24	0.0
LVM-RS (ours)	**76.77**	**74.99**	**71.26**	**67.55**	**63.43**	**58.59**	**51.39**	0.0

Tab. C.5.2.: Certified accuracy for $\sigma = 0.25$ on CIFAR-10, for risk $\alpha = 1\text{e}{-3}$ and $n = 10^5$ samples.

Methods	**Certified accuracy (ε)**									
	0.0	0.14	0.2	0.25	0.3	0.4	0.5	0.6	0.75	0.8
Carlini et al., 2023	86.72	80.73	77.47	74.41	71.15	65.01	58.25	51.15	40.96	37.6
LVM-RS (ours)	**88.49**	**82.15**	**79.06**	**76.21**	**72.73**	**66.41**	**60.22**	**53.41**	**43.76**	**40.27**

Tab. C.5.3.: Certified accuracy for $\sigma = 0.5$ on CIFAR-10, for risk $\alpha = 1\text{e}{-3}$ and $n = 10^5$ samples.

Methods	**Certified accuracy (ε)**									
	0.0	0.14	0.2	0.25	0.3	0.4	0.5	0.6	0.75	0.8
Carlini et al., 2023	74.11	67.99	65.22	62.89	60.38	55.67	50.43	45.59	39.26	37.11
LVM-RS (ours)	**79.79**	**73.45**	**70.41**	**68.04**	**65.8**	**60.71**	**55.48**	**50.07**	**43.13**	**40.83**

Tab. C.5.4.: Certified accuracy for $\sigma = 1$ on CIFAR-10, for risk $\alpha = 1e{-}3$ and $n = 10^5$ samples.

Methods	Certified accuracy (ε)									
	0.0	0.14	0.2	0.25	0.3	0.4	0.5	0.6	0.75	
Carlini et al., 2023	48.97	44.24	42.26	40.76	39.15	35.91	33.08	29.92	25.97	2
LVM-RS (ours)	**63.72**	**57.99**	**55.54**	**53.4**	**51.23**	**47.19**	**43.19**	**39.76**	**34.27**	**3**

Tab. C.5.5.: Certified accuracy for $\sigma = 0.25$ on ImageNet, for risk $\alpha = 1e{-}3$ and $n = 10^4$ samples.

Methods	Certified accuracy (ε)					
	0.0	0.5	1.0	1.5	2	3
Carlini et al., 2023	79.88	69.57	0.0	0.0	0.0	0.0
LVM-RS (ours)	**80.66**	**69.84**	0.0	0.0	0.0	0.0

Tab. C.5.6.: Certified accuracy for $\sigma = 0.5$ on ImageNet, for risk $\alpha = 1e{-}3$ and $n = 10^4$ samples.

Methods	Certified accuracy (ε)					
	0.0	0.5	1.0	1.5	2	3
Carlini et al., 2023	74.37	64.56	51.55	**36.04**	0.0	0.0
LVM-RS (ours)	**78.18**	**66.47**	**53.85**	**36.04**	0.0	0.0

Tab. C.5.7.: Certified accuracy for $\sigma = 1$ on ImageNet, for risk $\alpha = 1e{-}3$ and $n = 10^4$ samples.

Methods	Certified accuracy (ε)					
	0.0	0.5	1.0	1.5	2	3
Carlini et al., 2023	57.06	49.05	39.74	32.33	25.53	14.01
LVM-RS (ours)	**66.87**	**55.56**	**44.74**	**34.83**	**27.43**	**14.31**

D Regularization

D.1 Additional experiments

D.1.1 Activation Decay Faithfully Approximates Monte Carlo Smoothing of the Last Layer

Monte Carlo smoothing offers a way to flatten sharp minima by averaging the loss over random perturbations. However, it requires multiple stochastic forward-backward passes, incurring high computational cost. Activation Decay provides a deterministic and efficient surrogate based on a bound (obtained with Jensen's inequality) of the smoothed loss restricted to the final layer.

We train a 3-layer MLP with 512 hidden units and GELU activations on CIFAR-10 for 10 epochs using SGD (learning rate 0.1, cosine annealing, no momentum, no weight decay) and batch size 128. At each iteration, for the same mini-batch, we compute:

- the *Monte Carlo smoothed loss*:

$$\mathcal{L}^{\sigma}\left(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}\right)=\mathbb{E}_{\mathbf{\Delta}\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\left[\mathcal{L}\left((\mathbf{W}^{(L)}+\mathbf{\Delta})\mathbf{h}^{(L-1)},\mathbf{y}\right)\right]$$

  estimated via $n_{\text{MC}}=100$ samples, along with its gradient $\nabla_{\mathbf{W}^{(L)}}\mathcal{L}^{\sigma}$;

- the *Activation Decay surrogate loss*:

$$\tilde{\mathcal{L}}\left(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}\right)=\mathcal{L}\left(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}\right)+\frac{\sigma^2}{2}\left\|\mathbf{h}^{(L-1)}\right\|^2$$

  with corresponding gradient $\nabla_{\mathbf{W}^{(L)}}\tilde{\mathcal{L}}$.

For each value of $\sigma\in\{0.05,0.10,\ldots,1.00\}$, we log the following diagnostics, averaged over each epoch: cosine similarity between the two gradients, norm ratio $\|\nabla_{\text{MC}}\|_2/\|\nabla_{\text{AD}}\|_2$, and the loss gap $\mathcal{L}^{\sigma}-\tilde{\mathcal{L}}$.

σ	Cosine ↑ (MC, AD)	Norm ratio $\|\nabla_{\text{MC}}\|/\|\nabla_{\text{AD}}\|$	Loss gap $\mathcal{L}^{\sigma} - \tilde{\mathcal{L}}$
0.05	0.916	1.085	0.021
0.10	0.938	1.065	0.009
0.15	0.939	1.064	0.004
0.20	0.941	1.062	−0.001
0.25	0.938	1.068	−0.003
0.30	0.939	1.068	−0.006
0.35	0.935	1.071	−0.007
0.40	0.931	1.072	−0.007
0.50	0.933	1.078	−0.009
1.00	0.936	1.075	−0.009

Tab. D.1.1.: Comparison between Monte Carlo smoothing and Activation Decay surrogate. Values are averaged over ten training epochs. Cosine similarity close to 1 and near-unit norm ratios confirm gradient alignment; loss gaps remain below 1% of total loss.

The empirical results, see Table D.1.1, confirm the tight match between the gradients and loss values of the smoothed objective and its Activation Decay surrogate across a wide range of σ. Even at higher smoothing levels, gradients remain highly aligned (cosine ≥ 0.93), and loss discrepancies are negligible. This validates Activation Decay as an efficient and reliable approximation to Monte Carlo smoothing for last-layer perturbations.

D.1.2 Classification with MLP on CIFAR-10

In this experiment we use the same setting as in experiment 7.2.4. We conducted a comparison of our method AD, with the approach outlined in (Baek et al., 2024) Section 4.3, whose framework, designed specifically for label noise robustness, requires tuning decay parameters for each layer to achieve effective regularization. This introduces additional hyperparameter complexity but shares conceptual similarities with our approach.

To ensure a fair comparison, both methods were evaluated under identical experimental settings, including hyperparameter tuning for Baek et al., 2024's regularization coefficients. The results are summarized in Table D.1.2.

Tab. D.1.2.: Comparison with (Baek et al., 2024) and Activation Decay (AD) under identical experimental settings.

Metric	**(Baek et al., 2024)**	**AD (ours)**
Mean test accuracy (%)	65.05	65.11
95% confidence interval	[64.86, 65.23]	[64.90, 65.33]

Our findings demonstrate that AD, which applies ℓ_2-regularization to the penultimate activations, achieves comparable results to (Baek et al., 2024)'s method without the need for layer-wise tuning. Specifically, (Baek et al., 2024)'s method requires tuning separate coefficients for intermediate layers ($\sigma = 1\text{e}{-3}$) and for the last layer parameters ($\sigma = 1\text{e}{-2}$). In contrast, our approach eliminates this complexity and achieves similar performance with a single hyperparameter, σ, set to 0.1.

The simplicity of our method reduces the number of hyperparameters to tune, making it both more practical and easier to analyze. Furthermore, regularizing the penultimate activation indirectly regularizes preceding layers, as the penultimate activation encapsulates their contributions.

D.1.3 Classification with MLP-Mixer on ImageNet-1k

We test our method on the MLP-Mixer architecture (Tolstikhin et al., 2021), using the same settings as Liu et al., 2023 and Liu et al., 2022. We applied AD to the final layer and the cross-entropy loss of the Mixer-S/32 architecture on ImageNet-1k and reported the accuracy. Figure D.1.1 results indicate that AD improves model performance, leading to higher accuracy than the baseline.

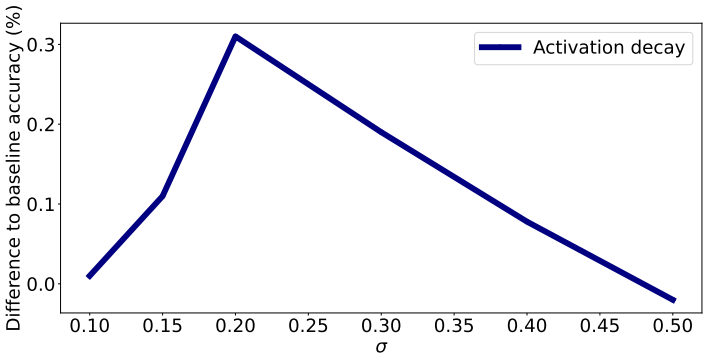


Fig. D.1.1.: Plot of different pieces of training on ImageNet of MLP-Mixer with AD, with varying σ.

D.1.4 Experiments on LLM

Tab. D.1.3.: Evaluation results for RoBERTa baseline and AD ($\sigma = 0.05$), on 7 tasks.

Metric	DO	AD
Sentiment Evaluation		
Classification Accuracy (%)	77.66	**77.68**
NER Evaluation		
Snips F1 Score (%)	72.70	**73.99**
Snips Precision (%)	67.78	**69.25**
Snips Recall (%)	78.39	**79.44**
Intent Evaluation		
Classification Accuracy (%)	**97.59**	96.83
Entailment SNLI Evaluation		
Classification Accuracy (%)	88.60	**89.33**
CoNLL NER Evaluation		
Seqeval F1 Score (%)	67.62	**67.90**
Seqeval Precision (%)	65.08	**65.38**
Seqeval Recall (%)	70.36	**70.61**
CoNLL POS Evaluation		
Seqeval F1 Score (%)	71.86	**72.14**
Seqeval Precision (%)	70.68	**70.89**
Seqeval Recall (%)	73.07	**73.45**
Query Correctness Evaluation		
Classification Accuracy (%)	**68.08**	68.06

D.2 Proofs

D.2.1 Proof of Corollary 7.2.2

Proof. Let $F(\theta) := \nabla_\theta \mathcal{L}(\theta) \in \mathbb{R}^d$. By assumption, we have

$$\|F(\theta)\|_2 \le \epsilon, \qquad \|F(\theta) - F(\theta')\|_2 \le H\|\theta - \theta'\|_2 \quad \text{for all } \theta, \theta' \in \mathbb{R}^d,$$

which is equivalent to $\|\nabla^2_\theta \mathcal{L}(\theta)\|_2 \le H$.
For any unit vector $u \in \mathbb{R}^d$, define

$$f_u(\theta) := u^\top F(\theta) + \epsilon.$$

Then $f_u : \mathbb{R}^d \to [0, 2\epsilon]$ is H-Lipschitz. Recall its Gaussian smoothed version as $f_u^\sigma(\theta) = \mathbb{E}_{\Delta \sim \mathcal{N}(0,\sigma^2 I)}[f_u(\theta + \Delta)]$.

Tab. D.1.4.: Evaluation results for BERT baseline with DO ($p = 0.1$), SAM for different ρ values, and AD with ($\sigma = 0.05$) on 7 tasks.

Metric	DO	SAM			AD
		$\rho = 0.01$	$\rho = 0.05$	$\rho = 0.1$	
Sentiment Evaluation					
Classification Accuracy (%)	76.72	76.54	75.38	62.28	**77.08**
NER Evaluation					
Snips F1 Score (%)	78.33	69.67	62.29	63.95	**80.90**
Snips Precision (%)	73.69	64.11	56.58	58.08	**76.20**
Snips Recall (%)	83.59	76.28	69.27	71.14	**86.21**
Intent Evaluation					
Classification Accuracy (%)	98.04	98.19	97.43	97.28	**98.49**
Entailment SNLI Evaluation					
Classification Accuracy (%)	87.96	**89.39**	86.77	83.85	88.88
CoNLL NER Evaluation					
Seqeval F1 Score (%)	64.43	61.01	51.09	52.05	**65.94**
Seqeval Precision (%)	61.87	61.48	51.64	51.72	**64.11**
Seqeval Recall (%)	67.20	60.55	50.56	52.39	**67.87**
CoNLL POS Evaluation					
Seqeval F1 Score (%)	75.95	72.48	69.31	71.20	**77.89**
Seqeval Precision (%)	74.89	71.59	68.79	70.46	**76.98**
Seqeval Recall (%)	77.04	73.39	69.84	71.97	**78.82**
Query Correctness Evaluation					
Classification Accuracy (%)	**69.95**	69.47	66.24	64.74	69.31

Applying Theorem **??** to f_u with $r = 2\epsilon$ and $L(f_u) = H$, we obtain

$$\mathrm{L}(f_u^\sigma) \leq H \operatorname{erf}\left(\frac{2\epsilon}{2^{3/2} H\sigma}\right) = H \operatorname{erf}\left(\frac{\epsilon}{\sqrt{2} H\sigma}\right). \tag{1}$$

Since $\mathcal{L}$ is twice differentiable, we can exchange the gradient and expectation:

$$\nabla_\theta \mathcal{L}^\sigma(\theta) = \mathbb{E}_\Delta[F(\theta + \Delta)].$$

Therefore,

$$\nabla_\theta f_u^\sigma(\theta) = \left(\nabla_\theta F^\sigma(\theta)\right)^\top u = \nabla_\theta^2 \mathcal{L}^\sigma(\theta)\, u.$$

Hence,

$$\mathrm{L}(f_u^\sigma) = \sup_\theta \left\| \nabla_\theta^2 \mathcal{L}^\sigma(\theta)\, u \right\|_2. \tag{2}$$

Combining (1) and (2), we obtain for all unit vectors u,

$$\sup_\theta \left\| \nabla_\theta^2 \mathcal{L}^\sigma(\theta)\, u \right\|_2 \leq H \operatorname{erf}\left(\frac{\epsilon}{\sqrt{2} H\sigma}\right).$$

Taking the supremum over all unit vectors u yields

$$\left\|\nabla_\theta^2 \mathcal{L}^\sigma(\theta)\right\|_2 \le H \operatorname{erf}\left(\frac{\epsilon}{\sqrt{2}H\sigma}\right),$$

which concludes the proof.

D.2.2 Proof of Theorem 7.2.1

Proof. Write $\mathbf{z} = \mathbf{h}^{(L-1)}$ and $J = \frac{\partial \mathbf{z}}{\partial \theta}$. By the second-order chain rule,

$$\nabla_\theta^2 \mathcal{L} = J^\top \nabla_{\mathbf{z}}^2 \mathcal{L}\, J \;+\; \left(\frac{\partial^2 \mathbf{z}}{\partial \theta^2}\right)^\top \nabla_{\mathbf{z}} \mathcal{L}.$$

Stationarity in logits implies $\nabla_{\mathbf{a}^{(L)}} \ell(\mathbf{a}^{(L)}(\theta^\star), y) = 0$ sample-wise, hence $\nabla_{\mathbf{z}} \mathcal{L}(\theta^\star) = \left(\frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{z}}\right)^\top \nabla_{\mathbf{a}^{(L)}} \ell(\mathbf{a}^{(L)}(\theta^\star), y) = 0$, So the residual term vanishes at $\theta^\star$ and

$$\nabla_\theta^2 \mathcal{L}(\theta^\star) = J(\theta^\star)^\top \nabla_{\mathbf{z}}^2 \mathcal{L}(\theta^\star)\, J(\theta^\star).$$

Taking operator norms and using submultiplicativity,

$$\|\nabla_\theta^2 \mathcal{L}(\theta^\star)\|_2 \;\le\; \|J(\theta^\star)\|_2^2 \, \|\nabla_{\mathbf{z}}^2 \mathcal{L}(\theta^\star)\|_2.$$

Finally, expand J layerwise:

$$\frac{\partial \mathbf{z}}{\partial \theta} = \sum_{j=1}^{L-1} \left(\prod_{l=j+1}^{L-1} \frac{\partial f^{(l)}}{\partial \mathbf{h}^{(l-1)}} \right) \frac{\partial \mathbf{h}^{(j)}}{\partial \theta},$$

whence, by the triangle inequality and the 1-Lipschitz property,

$$\|J(\theta^\star)\|_2 \;\le\; \sum_{j=1}^{L-1} \left\| \frac{\partial \mathbf{h}^{(j)}}{\partial \theta}(\theta^\star) \right\|_2 \prod_{l=j+1}^{L-1} \|\mathbf{W}^{(l)}\|_2.$$

Combine the two displays to obtain the claimed bound.

D.2.3 Proof of Theorem 7.2.3

Proof. The original cross-entropy loss for the correct class c is given by:

$$\mathcal{L}(h_L(\mathbf{h}_{L-1}), \mathbf{y}) = -\mathbf{W}_c^\top \mathbf{h}_{L-1} + \log\left(\sum_{j=1}^{d} \exp(\mathbf{W}_j^\top \mathbf{h}_{L-1})\right).$$

Consider the smoothed loss by introducing Gaussian noise $\boldsymbol{\Delta} \sim \mathcal{N}(0, \mathbf{I}\sigma^2)$:

$$\mathcal{L}^{\sigma}(h_L(\mathbf{h}_{L-1}), \mathbf{y}) = \mathbb{E}_{\boldsymbol{\Delta}\sim\mathcal{N}(0,\mathbf{I}\sigma^2)}\left[-(\mathbf{W}_c + \boldsymbol{\Delta}_c)^\top \mathbf{h}_{L-1} + \log\left(\sum_{j=1}^{d} \exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right)\right]$$

Separating the terms:

$$\mathcal{L}^{\sigma}(h_L(\mathbf{h}_{L-1}), \mathbf{y}) = -\mathbf{W}_c^\top \mathbf{h}_{L-1} + \mathbb{E}_{\boldsymbol{\Delta}\sim\mathcal{N}(0,\mathbf{I}\sigma^2)}\left[\log\left(\sum_{j=1}^{d} \exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right)\right].$$

By applying Jensen's inequality on the expectation inside the logarithm, we get:

$$\mathbb{E}_{\boldsymbol{\Delta}}\left[\log\left(\sum_{j=1}^{d} \exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right)\right] \leq \log\left(\mathbb{E}_{\boldsymbol{\Delta}}\left[\sum_{j=1}^{d} \exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right]\right).$$

Since $\boldsymbol{\Delta}_j \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, we use the moment generating function of the Gaussian distribution:

$$\mathbb{E}[e^Z] = e^{\mu + \frac{1}{2}\sigma^2}$$

Applying this to our case for each $\mathbf{W}_j^\top \mathbf{h}_{L-1} + \boldsymbol{\Delta}_j^\top \mathbf{h}_{L-1}$:

$$\mathbb{E}\left[\exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right] = \exp(\mathbf{W}_j^\top \mathbf{h}_{L-1})\mathbb{E}\left[\exp(\boldsymbol{\Delta}_j^\top \mathbf{h}_{L-1})\right]$$

Given $\boldsymbol{\Delta}_j \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ and $\boldsymbol{\Delta}_j^\top \mathbf{h}_{L-1}$ is a Gaussian with mean 0 and variance $\sigma^2\|\mathbf{h}_{L-1}\|^2$, we get:

$$\mathbb{E}\left[\exp(\boldsymbol{\Delta}_j^\top \mathbf{h}_{L-1})\right] = \exp\left(0 + \frac{1}{2}\sigma^2\|\mathbf{h}_{L-1}\|^2\right)$$

Therefore,

$$\mathbb{E}\left[\exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right] = \exp\left(\mathbf{W}_j^\top \mathbf{h}_{L-1} + \frac{1}{2}\sigma^2\|\mathbf{h}_{L-1}\|^2\right)$$

Summing over j:

$$\mathbb{E}_{\boldsymbol{\Delta}}\left[\sum_{j=1}^{d} \exp((\mathbf{W}_j + \boldsymbol{\Delta}_j)^\top \mathbf{h}_{L-1})\right] = \sum_{j=1}^{d} \exp\left(\mathbf{W}_j^\top \mathbf{h}_{L-1} + \frac{1}{2}\sigma^2\|\mathbf{h}_{L-1}\|^2\right)$$

Substituting this back into the expression for the smoothed loss, we obtain:

$$\mathcal{L}^{\sigma}(h_L(\mathbf{h}_{L-1}), \mathbf{y}) \leq -\mathbf{W}_c^\top \mathbf{h}_{L-1} + \log\left(\sum_{j=1}^{d} \exp\left(\mathbf{W}_j^\top \mathbf{h}_{L-1} + \frac{1}{2}\sigma^2\|\mathbf{h}_{L-1}\|^2\right)\right)$$

This result shows that the smoothed loss $\mathcal{L}^{\sigma}(h_L(\mathbf{h}_{L-1}), \mathbf{y})$ is bounded above by the original loss with an additional offset term $\frac{1}{2}\sigma^2\|\mathbf{h}_{L-1}\|^2$. This offset is akin to the

regularization term observed in the loss from the work of Tsuzuku et al. (2018) in term of $L\epsilon$ where L is the Lipschitz constant and ϵ the size of the perturbation.

D.2.4 Theorem and proof on tighter approximation using Taylor expansion

Theorem D.2.1 (Tighter approximation via Taylor expansion)**.** *Let $\mathbf{W}^{(L)} \in \mathbb{R}^{c\times d}$, $\mathbf{h}^{(L-1)} \in \mathbb{R}^{d}$, and $\mathbf{\Delta} \in \mathbb{R}^{c\times d}$ with elements drawn independently from $\mathcal{N}(0,\sigma^2)$. Denote $\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)})$. For small σ, the expected cross-entropy loss under perturbations $\mathbf{\Delta}$ is approximately:*

$$\mathbb{E}\Big[\mathcal{L}_{\mathrm{CE}}\big((\mathbf{W}^{(L)}+\mathbf{\Delta})\mathbf{h}^{(L-1)},\mathbf{y}\big)\Big] \approx \mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}) + \tfrac{1}{2}\sigma^2\|\mathbf{h}^{(L-1)}\|_2^2\sum_{i=1}^{c}\hat{y}_i(1-\hat{y}_i)\ .$$

Note that the obtained approximation of the smoothed loss is not an upper bound on the exact smoothed loss.

Proof. We start by expanding the cross-entropy loss around $\mathbf{W}^{(L)}\mathbf{h}^{(L-1)}$ using a first-order Taylor expansion:

$$\mathcal{L}_{\mathrm{CE}}\big((\mathbf{W}^{(L)}+\mathbf{\Delta})\mathbf{h}^{(L-1)},\mathbf{y}\big) \approx \mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}) + \nabla\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y})^{\top}(\mathbf{\Delta}\mathbf{h}^{(L-1)})\ ,$$

where $\nabla\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}) = \hat{\mathbf{y}} - \mathbf{y}$. The first-order term is then:

$$(\hat{\mathbf{y}}-\mathbf{y})^{\top}\mathbf{\Delta}\mathbf{h}^{(L-1)}\ .$$

Taking the expectation of this term with respect to $\mathbf{\Delta}$, we use the fact that $\mathbb{E}[\mathbf{\Delta}] = 0$, so the expectation of the first-order term is zero:

$$\mathbb{E}\Big[(\hat{\mathbf{y}}-\mathbf{y})^{\top}\mathbf{\Delta}\mathbf{h}^{(L-1)}\Big] = 0\ .$$

We then proceed with the second-order Taylor expansion:

$$\frac{1}{2}(\mathbf{\Delta}\mathbf{h}^{(L-1)})^{\top}\nabla^2\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y})(\mathbf{\Delta}\mathbf{h}^{(L-1)})\ ,$$

where the Hessian $\nabla^2\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y})$ is given by:

$$\nabla^2\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}) = \mathrm{diag}(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\hat{\mathbf{y}}^{\top}\ .$$

Now, we compute the expectation of the second-order term. Using the property of quadratic forms for Gaussian random variables, we have:

$$\mathbb{E}[\boldsymbol{\Delta}\mathbf{h}^{(L-1)}(\boldsymbol{\Delta}\mathbf{h}^{(L-1)})^\top] = \sigma^2\|\mathbf{h}^{(L-1)}\|_2^2\mathbf{I}_c \ ,$$

where $\mathbf{I}_c$ is the identity matrix in $\mathbb{R}^{c\times c}$. Thus, the second-order term simplifies to:

$$\frac{\sigma^2}{2}\|\mathbf{h}^{(L-1)}\|_2^2\mathrm{tr}\left(\nabla^2\mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y})\right) \ .$$

Finally, we compute the trace of the Hessian:

$$\mathrm{tr}\left(\mathrm{diag}(\hat{\mathbf{y}}) - \hat{\mathbf{y}}\hat{\mathbf{y}}^\top\right) = \sum_{i=1}^{c}\hat{y}_i(1-\hat{y}_i) \ ,$$

as the trace of $\hat{\mathbf{y}}\hat{\mathbf{y}}^\top$ is 1. Therefore, the second-order term becomes:

$$\frac{\sigma^2}{2}\|\mathbf{h}^{(L-1)}\|_2^2\sum_{i=1}^{c}\hat{y}_i(1-\hat{y}_i) \ .$$

Thus, the total approximation including both the first- and second-order terms is:

$$\mathbb{E}\left[\mathcal{L}_{\mathrm{CE}}\big((\mathbf{W}^{(L)}+\boldsymbol{\Delta})\mathbf{h}^{(L-1)},\mathbf{y}\big)\right] \approx \mathcal{L}_{\mathrm{CE}}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)},\mathbf{y}) + \frac{\sigma^2}{2}\|\mathbf{h}^{(L-1)}\|_2^2\sum_{i=1}^{c}\hat{y}_i(1-\hat{y}_i) \ .$$

Bibliography

Lyapunov, Aleksandr M. (1892). *The General Problem of the Stability of Motion*. Originally published in Russian, Kharkov, 1892. Taylor & Francis (Translated in 1992) (cit. on p. 3).

Tikhonov, Andrey N. (1943). "On the Stability of Inverse Problems". In: *Doklady Akademii Nauk SSSR*. In Russian (cit. on p. 3).

Cramér, Harald (1946). *Mathematical Methods of Statistics*. Princeton University Press (cit. on p. 3).

Von Neumann, John and Herman H. Goldstine (1947). "Numerical Inverting of Matrices of High Order". In: *Bulletin of the American Mathematical Society* (cit. on p. 3).

Lanczos, Cornelius (1950). "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *Journal of Research of the National Bureau of Standards* (cit. on pp. 45, 47, 48).

Arnoldi, Walter Edwin (1951). "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of applied mathematics* (cit. on pp. 45, 47, 48).

Dunn, Olive Jean (1961). *Multiple comparisons among means*. Stanford University Press (cit. on p. 117).

Kublanovskaya, V.N. (1962). "On some algorithms for the solution of the complete eigenvalue problem". In: *USSR Computational Mathematics and Mathematical Physics* (cit. on pp. 45, 48).

Pontryagin, Lev Semyonovich, Vladimir Grigorievich Boltyanskii, Revaz V. Gamkrelidze, and Evgenii Fedorovich Mishchenko (1962). *The Mathematical Theory of Optimal Processes*. Translated from the Russian by K. N. Trirogoff and edited by L. W. Neustadt. John Wiley & Sons (cit. on p. 168).

Polyak, Boris T (1964). "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* (cit. on p. 9).

Stein, Elias M. (1970). *Singular Integrals and Differentiability Properties of Functions*. Princeton, NJ: Princeton University Press (cit. on p. 30).

Björck, Åke and Clazett Bowie (1971). "An iterative algorithm for computing the best estimate of an orthogonal matrix". In: *SIAM Journal on Numerical Analysis* (cit. on p. 24).

Vapnik, Vladimir N. and A. Ya. Chervonenkis (1971). "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". In: *Theory of Probability and Its Applications* (cit. on p. 4).

Held, Michael, Philip Wolfe, and Henry P Crowder (1974). "Validation of subgradient optimization". In: *Mathematical Programming* 6.1, pp. 62–88 (cit. on p. 106).

Devroye, Luc and T. J. Wagner (1979a). "Distribution-free inequalities for the deleted and holdout error estimates". In: *IEEE Transactions on Information Theory* (cit. on p. 4).

– (1979b). "Distribution-free performance bounds for potential function rules". In: *IEEE Transactions on Information Theory* (cit. on p. 4).

Henderson, Harold V. and Shayle R. Searle (1981). "The Vec-Permutation Matrix, the Vec Operator and Kronecker Products: A Review". In: *Linear & Multilinear Algebra* (cit. on pp. 60, 216).

Stein, Charles M. (1981). "Estimation of the Mean of a Multivariate Normal Distribution". In: (cit. on p. 163).

Arkadi S. Nemirovsky, David B. Yudin (1983). *Problem Complexity and Method Efficiency in Optimization*. John Wiley & Sons (cit. on p. 3).

Valiant, Leslie G (1984). "A theory of the learnable". In: *Communications of the ACM* (cit. on p. 4).

Mark Jerrum Leslie G. Valiant, Vijay V. Vazirani (1986). "Random generation of combinatorial structures from a uniform distribution". In: *Theoretical Computer Science* (cit. on p. 3).

Hochberg, Yosef and Ajit C Tamhane (1987). *Multiple comparison procedures*. Wiley (cit. on p. 117).

Jain, Anil K (1989). *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall (cit. on pp. 56–58).

Friedland, Shmuel (1991). "Revisiting matrix squaring". In: *Linear algebra and its applications* (cit. on pp. 51, 53).

Drucker, Harris and Yann LeCun (1992). "Improving generalization performance using double backpropagation". In: *IEEE Transactions on Neural Networks* (cit. on p. 21).

Krogh, Anders and John A Hertz (1992). "A Simple Weight Decay can Improve Generalization". In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 6, 43, 135).

Benjamini, Yoav and Yosef Hochberg (1995). "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal Statistical Society: Series B (Methodological)* (cit. on p. 117).

Bishop, Chris M. (1995). "Training with Noise is Equivalent to Tikhonov Regularization". In: *Neural Computation* (cit. on pp. 43, 137, 138).

Bonnans, J. Frédéric and Alexander Shapiro (1996). *Perturbation Analysis of Optimization Problems*. New York: Springer-Verlag (cit. on p. 4).

Golub, G.H., C.F. Van Loan, C.F. Van Loan, and P.C.F. Van Loan (1996). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press (cit. on p. 47).

Lehoucq, Richard B et al. (1996). "Deflation techniques for an implicitly restarted Arnoldi iteration". In: *SIAM Journal on Matrix Analysis and Applications* (cit. on pp. 45, 48).

Zayed, Ahmed I. (1996). *Handbook of Function and Generalized Function Transformations*. CRC Press (cit. on pp. 29, 30).

Hochreiter, Sepp and Jürgen Schmidhuber (1997a). "Flat minima". In: *Neural computation* (cit. on pp. 5, 6, 40, 41, 43, 135).

– (1997b). "Long short-term memory". In: *Neural computation* (cit. on pp. 2, 8).

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* (cit. on pp. 1, 54).

Kearns, Michael and Dana Ron (1999). "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation". In: *Neural Computation* (cit. on p. 4).

McAllester, David A (1999). "Some PAC-Bayesian theorems". In: *Proceedings of the Conference on Computational Learning Theory (COLT)* (cit. on p. 4).

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999). *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab (cit. on p. 48).

Saltelli, Andrea, Karen Chan, and E. Marian Scott (2000). *Sensitivity Analysis*. John Wiley & Sons (cit. on p. 4).

Bousquet, Olivier and André Elisseeff (2002). "Stability and generalization". In: *Journal of machine learning research* (cit. on p. 4).

Nesterov, Yurii (2003). *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media (cit. on p. 5).

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer (cit. on p. 94).

Chellapilla, K., Siddharth Puri, and Patrice Simard (2006). "High Performance Convolutional Neural Networks for Document Processing". In: *Tenth International Workshop on Frontiers in Handwriting Recognition* (cit. on p. 55).

Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith (2006). "Calibrating noise to sensitivity in private data analysis". In: *Theory of Cryptography Conference* (cit. on p. 4).

Nocedal, Jorge and Stephen J. Wright (2006). *Numerical Optimization*. Springer (cit. on p. 48).

Massart, Pascal (2007). "Concentration inequalities and model selection". In: *École d'été de probabilités de Saint-Flour* (cit. on p. 125).

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and Computing* (cit. on p. 48).

Maurer, Andreas and Massimiliano Pontil (2009). "Empirical bernstein bounds and sample variance penalization". In: *Conference on Learning Theory* (cit. on pp. 110, 116, 125).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (cit. on p. 8).

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning* (ICML) (cit. on p. 8).

Duchi, John, Elad Hazan, and Yoram Singer (2011). “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research (JMLR)* (cit. on p. 9).

Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp (2011). “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”. In: *SIAM Review* (cit. on p. 48).

Zhang, Fuzhen (2011). *Matrix theory: basic results and techniques*. Springer Science & Business Media (cit. on p. 162).

Golub, Gene H. and Charles F. Van Loan (2012). *Matrix Computations*. Johns Hopkins University Press (cit. on p. 45).

Horn, R.A. and C.R. Johnson (2012). *Matrix Analysis*. Cambridge University Press (cit. on p. 50).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 1, 32, 54).

Tieleman, Tijmen and Geoffrey Hinton (2012). *Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude*. Coursera: Neural Networks for Machine Learning, `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf` (cit. on p. 9).

Boucheron, Stéphane, Gábor Lugosi, and Pascal Massart (2013). *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press (cit. on pp. 116, 124).

Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng (2013). “Rectifier Non-Linearities Improve Neural Network Acoustic Models”. In: *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (cit. on p. 8).

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). “On the difficulty of training recurrent neural networks”. In: *International Conference on Machine Learning* (cit. on pp. 8, 14).

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, et al. (2013). “Intriguing properties of neural networks”. In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 6, 14, 19, 32, 79, 97).

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, et al. (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *arXiv* (cit. on p. 2).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, et al. (2014). “Generative adversarial networks”. In: *Advances in Neural Information Processing Systems* (cit. on p. 2).

Kingma, Diederik P and Max Welling (2014). “Auto-encoding variational Bayes”. In: *International Conference on Learning Representations* (cit. on p. 2).

Simonyan, Karen and Andrew Zisserman (2014). “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv preprint arXiv:1409.1556* (cit. on p. 80).

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). “Dropout: A simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* (cit. on pp. 42, 137).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). *Neural Machine Translation by Jointly Learning to Align and Translate* (cit. on p. 2).

Choromanska, A., M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun (2015). "The loss surfaces of multilayer networks". In: *Artificial intelligence and statistics* (cit. on p. 41).

Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy (2015). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 6, 7, 14, 32).

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision* (ICCV) (cit. on p. 8).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning* (ICML) (cit. on pp. 8, 9, 20, 85, 86, 134, 138).

Kingma, Diederik P and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 9).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer (cit. on p. 1).

Abadi, Martín, Paul Barham, Jianmin Chen, et al. (2016). "TensorFlow: A system for large-scale machine learning". In: (cit. on p. 2).

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv* (cit. on pp. 8, 20, 86, 134).

Condat, Laurent (2016). "Fast projection onto the simplex and the ℓ_1 ball". In: *Mathematical Programming* 158.1, pp. 575–585 (cit. on p. 106).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press (cit. on p. 137).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (cit. on pp. 3, 8, 26, 72, 80, 82, 87, 98, 132, 138).

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian error linear units (GELUs)". In: *arXiv preprint arXiv:1606.08415* (cit. on p. 8).

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (2016). "Adversarial Examples in the Physical World". In: *arXiv preprint arXiv:1607.02533* (cit. on p. 7).

Mallat, Stéphane (2016). "Understanding deep convolutional networks". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (cit. on pp. 3, 54).

Martins, Andre and Ramon Astudillo (2016). "From softmax to sparsemax: A sparse model of attention and multi-label classification". In: *International Conference on Machine Learning* (cit. on pp. 106, 126).

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (cit. on p. 2).

Salimans, Tim and Diederik P. Kingma (2016). *Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks* (cit. on p. 21).

Silver, David, Aja Huang, Chris J Maddison, et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* (cit. on p. 1).

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 16, 21).

Balduzzi, David, Marcus Frean, Lennox Leary, et al. (2017). "The Shattered Gradients Problem: If ResNets are the Answer, Then What is the Question?" In: *Proceedings of the 34th International Conference on Machine Learning (ICML)* (cit. on pp. 8, 26).

Bartlett, Peter L, Dylan J Foster, and Matus J Telgarsky (2017). "Spectrally-normalized margin bounds for neural networks". In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 5, 14, 19, 40, 45, 79, 132, 144).

Bauschke, Heinz H. and Patrick L. Combettes (2017). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. 2nd. Springer (cit. on p. 27).

Carlini, Nicholas and David Wagner (2017). "Towards evaluating the robustness of neural networks". In: *IEEE Symposium on Security and Privacy* (cit. on pp. 7, 32).

Cisse, Moustapha, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier (2017). "Parseval Networks: Improving Robustness to Adversarial Examples". In: *Proceedings of the 34th International Conference on Machine Learning* (ICML) (cit. on pp. 7, 33).

Conneau, Alexis, Holger Schwenk, Loïc Barrault, and Yann Lecun (2017). "Very Deep Convolutional Networks for Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain (cit. on p. 55).

Dinh, L., R. Pascanu, S. Bengio, and Y. Bengio (2017). "Sharp minima can generalize for deep nets". In: *International Conference on Machine Learning* (cit. on p. 42).

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville (2017). "Improved training of wasserstein gans". In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 16, 21).

Jin, Chi, Rong Ge, Praneeth Netrapalli, Sham Kakade, and Michael I Jordan (2017). "How to Escape Saddle Points Efficiently". In: *International Conference on Machine Learning* (cit. on p. 42).

Katz, Guy, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer (2017). "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". In: *Computer Aided Verification*. Ed. by Rupak Majumdar and Viktor Kunčak (cit. on p. 97).

Keskar, Nitish Shirish, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang (2017). "ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA". In: (cit. on pp. 6, 40, 41).

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization* (cit. on p. 99).

Nesterov, Yurii and Vladimir Spokoiny (2017). "Random Gradient-Free Minimization of Convex Functions". In: *Foundations of Computational Mathematics* (cit. on pp. 31, 139).

Sokolić, Jure, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues (2017). "Robust Large Margin Deep Neural Networks". In: *IEEE Transactions on Signal Processing* (cit. on p. 5).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 2, 19, 151).

Yoshida, Yuichi and Takeru Miyato (2017). "Spectral Norm Regularization for Improving the Generalizability of Deep Learning". In: *arXiv preprint arXiv:1705.10941* (cit. on pp. 21, 25, 41, 81).

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals (2017). "Understanding deep learning requires rethinking generalization". In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 6, 42, 135, 136).

Athalye, Anish, Nicholas Carlini, and David Wagner (2018). "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning* (ICML) (cit. on p. 33).

Elsayed, Gamaleldin, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio (2018). "Large margin deep networks for classification". In: *Advances in neural information processing systems* 31 (cit. on p. 43).

Finlay, Chris, Jeff Calder, Bilal Abbasi, and Adam Oberman (2018). "Lipschitz Regularized Deep Neural Networks Generalize and are Adversarially Robust". In: *arXiv preprint arXiv:1808.09540* (cit. on p. 41).

Gao, Bolin and Lacra Pavel (2018). "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning". In: (cit. on p. 19).

Haber, Eldad and Lars Ruthotto (2018). "Stable architectures for deep neural networks". In: *Inverse Problems* (cit. on p. 11).

Kodali, Naveen, James Hays, Jacob Abernethy, and Zsolt Kira (2018). "On Convergence and Stability of GANs". In: *ICLR (Workshop or Blind Sub. if applicable)* (cit. on p. 98).

Laha, Anirban, Saneem Ahmed Chemmengath, Priyanka Agrawal, et al. (2018). "On Controllable Sparse Alternatives to Softmax". In: *Advances in Neural Information Processing Systems* (cit. on p. 106).

Lecuyer, M., V. Atlidakais, R. Geambasu, D. Hsu, and S. Jana (2018). "Certified Robustness to Adversarial Examples with Differential Privacy". In: *2019 IEEE Symposium on Security and Privacy (SP)* (cit. on pp. 7, 29, 33).

Li, Bai, Changyou Chen, Wenlin Wang, and Lawrence Carin (2018a). "Second-Order Adversarial Attack and Certifiable Robustness". In: *arXiv preprint arXiv:1809.03113* (cit. on p. 33).

– (2018b). "Second-Order Adversarial Attack and Certifiable Robustness". In: *International Conference on Learning Representations* (cit. on p. 38).

Li, Baoyuan, Changyou Chen, Wenlin Wang, and Lawrence Carin (2018c). "Second-order adversarial attack and certifiable robustness". In: *arXiv* (cit. on p. 7).

Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu (2018). "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 7, 32, 33, 110).

Miyato, Takeru, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida (2018). “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 9, 22, 25, 45, 81, 88).

Neyshabur, Behnam, Srinadh Bhojanapalli, and Nathan Srebro (2018). “A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks”. In: *International Conference on Learning Representations* (ICLR) (cit. on p. 5).

Raghunathan, Aditi, Jacob Steinhardt, and Percy Liang (2018). “Certified defenses against adversarial examples”. In: *International Conference on Learning Representations (ICLR)* (cit. on p. 7).

Ross, Andrew Slavin and Finale Doshi-Velez (2018). “Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (cit. on p. 21).

Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry (2018). “How does batch normalization help optimization?” In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on pp. 20, 87).

Shi, Guanya, Xiyang Shi, Michael O’Connell, et al. (2018). “Neural Lander: Stable Drone Landing Control Using Learned Dynamics”. In: *2019 International Conference on Robotics and Automation (ICRA)* (cit. on p. 16).

Tsuzuku, Yusuke, Issei Sato, and Masashi Sugiyama (2018). “Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 7, 14, 19, 22, 33–36, 45, 54, 79, 97, 99, 100, 104, 123, 194).

Virmaux, Aladin and Kevin Scaman (2018). “Lipschitz regularity of deep neural networks: analysis and efficient estimation”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 14, 17, 41, 45).

Wong, Eric and Zico Kolter (2018). “Provable defenses against adversarial examples via the convex outer adversarial polytope”. In: *Proceedings of the 35th International Conference on Machine Learning* (ICML) (cit. on p. 7).

Wong, Eric, Ludwig Schmidt, and J Zico Kolter (2018). “Scaling provable adversarial defenses”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (cit. on p. 15).

Wu, Yuxin and Kaiming He (2018). “Group normalization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on pp. 8, 20).

Xing, C., D. Arpit, C. Tsirigotis, and Y. Bengio (2018). “A Walk with SGD”. In: *arXiv* (cit. on p. 41).

Zhang, Huishuai, Wei Chen, and Tie-Yan Liu (2018). “On the Local Hessian in Back-propagation”. In: *Conference on Neural Information Processing Systems* (cit. on p. 138).

Anil, Cem, James Lucas, and Roger Grosse (2019). “Sorting out Lipschitz Function Approximation”. In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (cit. on pp. 7, 23, 25, 26, 98).

Behrmann, Jens, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen (2019). “Invertible Residual Networks”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)* (cit. on p. 77).

Bibi, Adel, Bernard Ghanem, Vladlen Koltun, and Rene Ranftl (2019). “Deep Layers as Stochastic Solvers”. In: *International Conference on Learning Representations* (ICLR) (cit. on p. 56).

Chaudhari, P., A. Choromanska, S. Soatto, et al. (2019). “Entropy-sgd: Biasing gradient descent into wide valleys”. In: *Journal of Statistical Mechanics: Theory and Experiment* (cit. on p. 40).

Cohen, Jeremy, Elan Rosenfeld, and Zico Kolter (2019). “Certified Adversarial Robustness via Randomized Smoothing”. In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (cit. on pp. 7, 29, 31, 33, 38–40, 85, 97, 100, 102, 105, 110, 116, 123–125).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL) (cit. on pp. 2, 145).

Falk, Thorsten, Dominic Mai, Robert Bensch, et al. (2019). “U-Net: deep learning for cell counting, detection, and morphometry”. In: *Nature Methods* (cit. on p. 1).

Farnia, Farzan, Jesse Zhang, and David Tse (2019a). “Generalizable Adversarial Training via Spectral Normalization”. In: *International Conference on Learning Representations* (ICLR) (cit. on p. 25).

Farnia, Farzan, Jesse M Zhang, and David N Tse (2019b). “Generalizable Adversarial Training via Spectral Normalization”. In: (cit. on pp. 25, 56).

Fazlyab, Mahyar, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas (2019). “Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 14, 17, 22, 26).

Huang, Po-Sen, He He, Anshul Chen, et al. (2019). “Achieving certified robustness to word substitutions with differential privacy”. In: *Annual Meeting of the Association for Computational Linguistics (ACL)* (cit. on p. 152).

Islam, Md Amirul, Sen Jia, and Neil D. B. Bruce (2019). “How much Position Information Do Convolutional Neural Networks Encode?” In: *International Conference on Learning Representations* (ICLR) (cit. on p. 55).

Jia, Robin and Percy Liang (2019). “Certified robustness to adversarial word substitutions”. In: *Empirical Methods in Natural Language Processing (EMNLP)* (cit. on p. 152).

Lecuyer, Mathias, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana (2019). “Certified robustness to adversarial examples with differential privacy”. In: *IEEE symposium on security and privacy (SP)* (cit. on pp. 38, 40, 125).

Levine, Alexander, Sahil Singla, and Soheil Feizi (2019). “Certifiably Robust Interpretation in Deep Learning”. In: (cit. on pp. 37, 40, 123, 125).

Li, Qiyang, Saminul Haque, Cem Anil, et al. (2019). “Preventing Gradient Attenuation in Lipschitz Constrained Convolutional Networks”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 7, 24, 65).

Liu, Yinhan, Myle Ott, Naman Goyal, et al. (2019). “RoBERTa: A robustly optimized BERT pretraining approach”. In: *arXiv* (cit. on p. 145).

Loshchilov, Ilya and Frank Hutter (2019). “Decoupled weight decay regularization”. In: *International Conference on Learning Representations (ICLR)* (cit. on p. 9).

Lugosi, Gábor and Shahar Mendelson (2019). “Mean Estimation and Regression Under Heavy-Tailed Distributions: A Survey”. In: *Foundations of Computational Mathematics* (cit. on p. 3).

Paszke, Adam, Sam Gross, Francisco Massa, et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on p. 2).

Pfister, Luke and Yoram Bresler (2019). “Bounding multivariate trigonometric polynomials”. In: *IEEE Transactions on Signal Processing* (cit. on pp. 66, 156).

Pinot, Rafael, Laurent Meunier, Alexandre Araujo, et al. (2019). “Theoretical Evidence for Adversarial Robustness through Randomization”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on p. 7).

Ryu, Ernest, Jialin Liu, Sicheng Wang, et al. (2019). “Plug-and-Play Methods Provably Converge with Properly Trained Denoisers”. In: *Proceedings of the 36th International Conference on Machine Learning* (ICML) (cit. on pp. 56, 57, 63, 71–75, 80, 81, 132–134, 159, 161, 217, 223, 225).

Salman, Hadi, Jerry Li, Ilya Razenshteyn, et al. (2019). “Provably robust deep learning via adversarially trained smoothed classifiers”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 7, 10, 29, 31, 38, 39, 91, 92, 94, 101, 102, 105, 111, 124, 166, 184).

Sedghi, Hanie, Vineet Gupta, and Philip Long (2019). “The Singular Values of Convolutional Layers”. In: *International Conference on Learning Representations* (cit. on pp. 54, 56, 59, 60, 71–74, 80–82, 133, 134, 159, 217, 223, 225).

Stickland, Asa Cooper and Iain Murray (2019). “Bert and pals: Projected attention layers for efficient adaptation in multi-task learning”. In: *International Conference on Machine Learning* (cit. on p. 145).

Zhang, Huan, Pengchuan Zhang, and Cho-Jui Hsieh (2019a). “RecurJac: An Efficient Recursive Algorithm for Bounding Jacobian Matrix of Neural Networks and Its Applications”. In: *AAAI Conference on Artificial Intelligence* (cit. on p. 17).

Zhang, Richard (2019). “Making Convolutional Networks Shift-Invariant Again”. In: *International Conference on Machine Learning* (cit. on p. 55).

Zhang, Si, Hanghang Tong, Jiejun Xu, and Ross Maciejewski (2019b). “Graph convolutional networks: a comprehensive review”. In: *Computational Social Networks* (cit. on p. 55).

al., Glenn Jocher et (2020). *YOLOv5* (cit. on p. 2).

Araujo, Alexandre, Benjamin Negrevergne, Yann Chevaleyre, and Jamal Atif (Nov. 2020). “On Lipschitz Regularization of Convolutional Layers using Toeplitz Matrix Theory”. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. arXiv: 2006.08391 (cit. on p. 132).

Baevski, Alexei, Henry Zhou, Abdelrahman Mohamed, and Michael Auli (2020). *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations* (cit. on p. 55).

Brown, Tom B., Benjamin Mann, Nick Ryder, et al. (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems* (NeurIPS) (cit. on pp. 2, 145).

Croce, Francesco and Matthias Hein (2020). “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks”. In: *International Conference on Machine Learning (ICML)* (cit. on p. 7).

Haldar, Siddhant (2020). *Gradient-based Adversarial Attacks: An Introduction*. `https://medium.com/swlh/gradient-based-adversarial-attacks-an-introduction-123abc456def` (cit. on pp. 33, 215).

Hendrycks, Dan, Collin Burns, Steven Basart, et al. (2020). “Measuring Massive Multitask Language Understanding”. In: *arXiv* (cit. on p. 147).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* (cit. on p. 2).

Huang, Lei, Li Liu, Fan Zhu, et al. (2020). “Controllable Orthogonalization in Training DNNs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (cit. on p. 21).

Jiang*, Yiding, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio (2020). “Fantastic Generalization Measures and Where to Find Them”. In: *International Conference on Learning Representations* (cit. on pp. 42, 135).

Jordan, Matt and Alexandros G Dimakis (2020). “Exactly Computing the Local Lipschitz Constant of ReLU Networks”. In: *Advances in Neural Information Processing Systems* (cit. on p. 17).

Kaplan, Jared, Sam McCandlish, Tom Henighan, et al. (2020). “Scaling Laws for Neural Language Models”. In: *arXiv preprint arXiv:2001.08361* (cit. on p. 2).

Latorre, Fabian, Paul Rolland, and Volkan Cevher (2020). “Lipschitz Constant Estimation for Neural Networks via Sparse Polynomial Optimization”. In: *International Conference on Learning Representations* (ICLR) (cit. on p. 17).

Li, Mingjie, Lingshen He, and Zhouchen Lin (2020). “Implicit Euler skip connections: Enhancing adversarial robustness via numerical stability”. In: *International Conference on Machine Learning (ICML)* (cit. on p. 7).

Raffel, Colin, Noam Shazeer, Adam Roberts, et al. (2020). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* (cit. on pp. 145, 147).

Salman, Hadi, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter (2020). “Denoised smoothing: A provable defense for pretrained classifiers”. In: *Advances in Neural Information Processing Systems* (cit. on p. 124).

Semih Kayhan, Osman and Jan C. Van Gemert (2020). “On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location”. In: *Computer Vision and Pattern Recognition* (cit. on p. 55).

Wang, Jiayun, Yubei Chen, Rudrasis Chakraborty, and Stella X. Yu (2020). “Orthogonal Convolutional Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (cit. on pp. 21, 62, 81).

Wei, Colin, Samuel S. Schoenholz, Zhi Dai, and Tengyu Ma (2020). “The Implicit and Explicit Regularization Effects of Dropout”. In: *International Conference on Machine Learning* (cit. on p. 137).

Yang, Greg, Tony Duan, J. Edward Hu, et al. (2020a). “Randomized Smoothing of All Shapes and Sizes”. In: *Proceedings of the 37th International Conference on Machine Learning* (cit. on p. 39).

Yang, Yao-Yuan, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri (2020b). “A Closer Look at Accuracy vs. Robustness”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (cit. on pp. 16, 18).

Yao, Zhewei, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W. Mahoney (2020). “PyHessian: Neural Networks through the Lens of the Hessian”. In: *IEEE International Conference on Big Data* (cit. on p. 139).

Yi, Xinping (2020). “Asymptotic Singular Value Distribution of Linear Convolutional Layers”. In: *arXiv preprint arXiv:2006.07117* (cit. on pp. 56, 57, 59, 60, 158).

You, Yang, Jingyu Li, Sashank J Reddi, et al. (2020). “Large batch optimization for deep learning: Training bert in 76 minutes”. In: *arXiv* (cit. on p. 9).

Araujo, Alexandre, Benjamin Negrevergne, Yann Chevaleyre, and Jamal Atif (2021). “On Lipschitz Regularization of Convolutional Layers using Toeplitz Matrix Theory”. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (cit. on pp. 56, 57, 66, 71, 75, 81, 82, 84, 132–134, 217, 218, 223).

Bubeck, Sébastien, Yuanzhi Li, and Dheeraj Nagaraj (2021). “A Law of Robustness for Two-Layers Neural Networks”. In: *Proceedings of the 34th Annual Conference on Learning Theory*. PMLR (cit. on pp. 16, 150).

Bubeck, Sebastien and Mark Sellke (2021). “A Universal Law of Robustness via Isoperimetry”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (cit. on pp. 16, 150).

Croce, Francesco, Maksym Andriushchenko, Vikash Sehwag, et al. (2021). “RobustBench: a standardized adversarial robustness benchmark”. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)* (cit. on p. 97).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, et al. (2021). “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 2, 3).

Foret, Pierre, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur (2021). “Sharpness-aware Minimization for Efficiently Improving Generalization”. In: *International Conference on Learning Representations* (cit. on pp. 6, 42, 138, 142, 144).

Gouk, Henry, Eibe Frank, Bernhard Pfahringer, and Michael J Cree (2021). “Regularisation of neural networks by enforcing lipschitz continuity”. In: *Machine Learning* (cit. on pp. 15, 22, 41, 81).

Huang, Yujia, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar (2021). “Training Certifiably Robust Neural Networks with Efficient Local Lipschitz Bounds”. In: (cit. on p. 26).

Jumper, John, Richard Evans, Alexander Pritzel, et al. (2021). “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* (cit. on p. 1).

Kim, Hyunjik, George Papamakarios, and Andriy Mnih (2021). *The Lipschitz Constant of Self-Attention* (cit. on pp. 19, 29).

Kwon, Jungmin, Hae Beom Park, Junho Kim, and Il Dong Choi (2021). “ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks”. In: *Advances in Neural Information Processing Systems* (cit. on p. 143).

Leino, Klas and Matt Fredrikson (2021). “Relaxing Local Robustness”. In: *Neural Information Processing Systems (NIPS)* (cit. on p. 22).

Leino, Klas, Zifan Wang, and Matt Fredrikson (2021). “Globally-Robust Neural Networks”. In: *International Conference on Machine Learning (ICML)* (cit. on p. 22).

Lu, Zhiyun, Eugene Ie, and Fei Sha (2021). “Mean-Field Approximation to Gaussian-Softmax Integral with Application to Uncertainty Estimation”. In: *arXiv* (cit. on p. 140).

Newman, Elizabeth, Lars Ruthotto, Joseph Hart, and Bart van Bloemen Waanders (2021). “Train Like a (Var)Pro: Efficient Training of Neural Networks with Variable Projection”. In: *SIAM Journal on Mathematics of Data Science* (cit. on p. 140).

Singla, Sahil and Soheil Feizi (2021a). “Fantastic Four: Differentiable and Efficient Bounds on Singular Values of Convolution Layers”. In: *International Conference on Learning Representations* (cit. on pp. 56, 57, 71, 72, 80–82, 84, 132–134, 159, 218, 223, 225).

– (2021b). “Improved Lipschitz bounds for deep networks: Scaling law and effective approximation”. In: *International Conference on Learning Representations (ICLR)* (cit. on p. 7).

– (2021c). “Skew Orthogonal Convolutions”. In: *Proceedings of the 38th International Conference on Machine Learning* (cit. on pp. 7, 24, 27, 65).

Tan, Mingxing and Quoc Le (2021). “EfficientNetV2: Smaller Models and Faster Training”. In: *International Conference on Machine Learning* (cit. on p. 54).

Tolstikhin, Ilya, Neil Houlsby, Alexander Kolesnikov, et al. (2021). “MLP-Mixer: An all-MLP Architecture for Vision”. In: *Advances in Neural Information Processing Systems* (cit. on pp. 142, 189).

Trockman, Asher and J Zico Kolter (2021). “Orthogonalizing Convolutional Layers with the Cayley Transform”. In: *International Conference on Learning Representations* (cit. on pp. 7, 19, 24, 26, 27, 59, 65).

Vuckovic, James, Aristide Baratin, and Remi Tachet des Combes (2021). *On the Regularity of Attention* (cit. on p. 19).

Zhang, Bo, Tianle Cai, Ziyu Lu, Di He, and Liwei Wang (2021a). “Towards certifying ℓ_∞ robustness using neural networks with ℓ_∞-dist neurons”. In: *International Conference on Machine Learning* (cit. on p. 15).

Zhang, Zhihan, Wenhao Yu, Mengxia Yu, Zhichun Guo, and Meng Jiang (2021b). “A Survey of Multi-task Learning in Natural Language Processing: Regarding Task Relatedness and Training Methods”. In: *arXiv* (cit. on p. 144).

Ablin, Pierre and Gabriel Peyré (2022). “Fast and accurate optimization on the orthogonal manifold without retraction”. In: *International Conference on Artificial Intelligence and Statistics* (cit. on p. 23).

Achour, El Mehdi, François Malgouyres, and Franck Mamalet (2022). “Existence, Stability and Scalability of Orthogonal Convolutional Neural Networks”. In: *Journal of Machine Learning Research* (cit. on p. 81).

Ahmad, Kashif, Majdi Maabreh, Mohamed Ghaly, et al. (2022). “Developing future human-centered smart cities: Critical analysis of smart city security, Data management, and Ethical challenges”. In: *Computer Science Review* (cit. on pp. 7, 215).

Andriushchenko, Maksym and Nicolas Flammarion (2022). “Towards Understanding Sharpness-Aware Minimization”. In: *Proceedings of the 39th International Conference on Machine Learning* (cit. on pp. 6, 42).

Béthune, Louis, Thibaut Boissin, Mathieu Serrurier, et al. (2022). “Pay attention to your loss : understanding misconceptions about Lipschitz neural networks”. In: *Advances in Neural Information Processing Systems* (cit. on p. 96).

Brunke, Lukas, Michael Greeff, Andrew W Hall, et al. (2022). “Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* (cit. on p. 16).

Gu, Albert, Karan Goel, and Christopher Re (2022). “Efficiently Modeling Long Sequences with Structured State Spaces”. In: *International Conference on Learning Representations* (cit. on p. 55).

Li, Zewen, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou (2022). “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Trans Neural Netw Learn Syst* (cit. on p. 55).

Liu, Zhuang, Hanzi Mao, Chao-Yuan Wu, et al. (2022). “A ConvNet for the 2020s”. In: *Conference on Computer Vision and Pattern Recognition* (cit. on pp. 2, 189).

Meunier, Laurent, Blaise J. Delattre, Alexandre Araujo, and Alexandre Allauzen (2022). “A Dynamical System Perspective for Lipschitz Neural Networks”. In: *Proceedings of the 39th International Conference on Machine Learning* (cit. on pp. 19, 26–28, 90, 150, 151).

Orvieto, Antonio, Hans Kersting, Frank Proske, Francis Bach, and Aurelien Lucchi (2022). “Anticorrelated Noise Injection for Improved Generalization”. In: *Proceedings of the 39th International Conference on Machine Learning* (cit. on pp. 42, 137–139).

Pautov, Mikhail, Olesya Kuznetsova, Nurislam Tursynbek, Aleksandr Petiushko, and Ivan Oseledets (2022). “Smoothed Embeddings for Certified Few-Shot Learning”. In: *Advances in Neural Information Processing Systems* (cit. on p. 92).

Prach, Bernd and Christoph H Lampert (2022). “Almost-orthogonal layers for efficient general-purpose Lipschitz networks”. In: *Computer Vision–ECCV 2022: 17th European Conference* (cit. on pp. 25, 62, 88, 90, 98, 99, 155).

Sanh, Victor, Albert Webson, Colin Raffel, et al. (2022). “Multitask Prompted Training Enables Zero-Shot Task Generalization”. In: *International Conference on Learning Representations* (cit. on p. 145).

Senderovich, Alexandra, Ekaterina Bulatova, Anton Obukhov, and Maxim Rakhuba (2022). "Towards Practical Control of Singular Values of Convolutional Layers". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 36th Conference on Neural Information Processing Systems (cit. on p. 68).

Shi, Zhouxing, Yihan Wang, Huan Zhang, J. Zico Kolter, and Cho-Jui Hsieh (2022). "Efficiently Computing Local Lipschitz Constants of Neural Networks via Bound Propagation". In: *Advances in Neural Information Processing Systems* (cit. on p. 17).

Singla, Sahil, Surbhi Singla, and Soheil Feizi (2022). "Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100". In: *International Conference on Learning Representations* (cit. on p. 26).

Wang, Wei, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann (2022). "Robust Differentiable SVD". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (cit. on p. 47).

Wei, Jason, Maarten Bosma, Vincent Zhao, et al. (2022a). "Finetuned Language Models are Zero-Shot Learners". In: *International Conference on Learning Representations* (cit. on p. 145).

Wei, Jason, Yi Tay, Rishi Bommasani, et al. (2022b). "Emergent Abilities of Large Language Models". In: *arXiv* (cit. on p. 2).

Wen, Kaiyue, Tengyu Ma, and Zhiyuan Li (2022). "How Does Sharpness-Aware Minimization Minimizes Sharpness?" In: *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)* (cit. on p. 144).

Yu, Tan, Jun Li, YUNFENG CAI, and Ping Li (2022). "Constructing Orthogonal Convolutions in an Explicit Manner". In: *International Conference on Learning Representations* (cit. on p. 65).

Zhang, Bo, Difan Jiang, Di He, and Liwei Wang (2022a). "Boosting the certified robustness of ℓ_∞ distance nets". In: *International Conference on Learning Representations* (cit. on p. 15).

Zhang, Bohang, Du Jiang, Di He, and Liwei Wang (2022b). "Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective". In: *Advances in Neural Information Processing Systems* (cit. on p. 15).

Araujo, Alexandre, Aaron J Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu (2023). "A Unified Algebraic Perspective on Lipschitz Neural Networks". In: *International Conference on Learning Representations* (ICLR) (cit. on pp. 19, 26–28, 89, 90, 98, 99, 150, 162).

Béthune, Louis, Paul Novello, Guillaume Coiffier, et al. (2023). "Robust One-Class Classification with Signed Distance Function using 1-Lipschitz Neural Networks". In: *Proceedings of the 40th International Conference on Machine Learning*. Honolulu, Hawaii, USA (cit. on p. 16).

Carlini, Nicholas, Florian Tramer, Krishnamurthy Dj Dvijotham, et al. (2023). "(Certified!!) Adversarial Robustness for Free!" In: *International Conference on Learning Representations* (cit. on pp. 124, 125, 128, 129, 150, 182–185).

Franco, Nicola, Daniel Korth, Jeanette Miriam Lorenz, Karsten Roscher, and Stephan Guennemann (2023). "Diffusion Denoised Smoothing for Certified and Adversarial Robust Out-Of-Distribution Detection". In: *arXiv* (cit. on p. 92).

Fu, Stephanie, Netanel Yakir Tamir, Shobhita Sundaram, et al. (2023). “DreamSim: Learning New Dimensions of Human Visual Similarity using Synthetic Data”. In: *Thirty-seventh Conference on Neural Information Processing Systems* (cit. on p. 16).

Gavrikov, Paul and Janis Keuper (2023). “On the Interplay of Convolutional Padding and Adversarial Robustness”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops* (cit. on p. 55).

Hu, Kai, Andy Zou, Zifan Wang, Klas Leino, and Matt Fredrikson (2023). “Unlocking Deterministic Robustness Certification on ImageNet”. In: *Conference on Neural Information Processing Systems* (cit. on pp. 28, 110, 150, 151).

Liu, Zhuang, Zhiqiu Xu, Joseph Jin, Zhiqiang Shen, and Trevor Darrell (2023). “Dropout Reduces Underfitting”. In: *International Conference on Machine Learning* (cit. on pp. 42, 189).

OpenAI (2023). *Simple Evals: OpenAI Evaluation Framework*. `https://github.com/openai/simple-evals` (cit. on p. 147).

Pal, Ambar and Jeremias Sulam (2023). “Understanding Noise-Augmented Training for Randomized Smoothing”. In: *Transactions on Machine Learning Research* (cit. on p. 124).

Pinson, Hannah, Joeri Lenaerts, and Vincent Ginis (2023). “Linear CNNs Discover the Statistical Structure of the Dataset Using Only the Most Dominant Frequencies”. In: *International Conference on Machine Learning* (cit. on p. 55).

Scholten, Yan, Jan Schuchardt, Aleksandar Bojchevski, and Stephan Günnemann (2023). “Hierarchical Randomized Smoothing”. In: *Conference on Neural Information Processing Systems* (cit. on pp. 117, 118).

Singh, Sidak Pal, Thomas Hofmann, and Bernhard Schölkopf (2023). “The Hessian perspective into the Nature of Convolutional Neural Networks”. In: *International Conference on Machine Learning* (cit. on p. 55).

Tang, Ling, Wen Shen, Zhanpeng Zhou, Yuefeng Chen, and Quanshi Zhang (2023). “Defects of Convolutional Decoder Networks in Frequency Representation”. In: *International Conference on Machine Learning* (cit. on p. 55).

Verine, A., B. Negrevergne, Y. Chevaleyre, and F. Rossi (2023). “On the expressivity of bi-Lipschitz normalizing flows”. In: *Proceedings of the Asian Conference on Machine Learning (ACML)* (cit. on p. 17).

Voráček, Václav and Matthias Hein (2023). “Improving l1-Certified Robustness via Randomized Smoothing by Leveraging Box Constraints”. In: *International Conference on Machine Learning* (cit. on pp. 114, 115, 117, 118).

Wang, Ruigang and Ian Manchester (2023). “Direct parameterization of lipschitz-bounded deep networks”. In: *International Conference on Machine Learning* (cit. on pp. 26, 108).

Wang, Zhen, Rameswar Panda, Leonid Karlinsky, et al. (2023). “Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning”. In: *International Conference on Learning Representations* (cit. on p. 145).

Weber, Maurice, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li (2023). “RAB: Provable Robustness Against Backdoor Attacks”. In: IEEE (cit. on pp. 117, 118).

Baek, Christina, J Zico Kolter, and Aditi Raghunathan (2024). “Why is SAM Robust to Label Noise?” In: *International Conference on Learning Representations* (cit. on pp. 138, 142, 144, 188, 189, 225).

Béthune, Louis (2024). “Deep learning with Lipschitz constraints”. PhD thesis. Université de Toulouse (cit. on pp. 16, 24).

Béthune, Louis, Thomas Massena, Thibaut Boissin, et al. (2024). “DP-SGD Without Clipping: The Lipschitz Neural Network Way”. In: *The Twelfth International Conference on Learning Representations* (cit. on p. 16).

Castin, Valérie, Pierre Ablin, and Gabriel Peyré (2024). “How Smooth Is Attention?” In: *Proceedings of the 41st International Conference on Machine Learning* (cit. on p. 19).

Chen, Huanran, Yinpeng Dong, Shitong Shao, et al. (2024a). “Diffusion Models are Certifiably Robust Classifiers”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems* (cit. on pp. 92, 102).

Chen, Huanran, Yinpeng Dong, Zhengyi Wang, et al. (2024b). *Robust Classification via a Single Diffusion Model* (cit. on p. 102).

Ghazanfari, Sara, Alexandre Araujo, Prashanth Krishnamurthy, Farshad Khorrami, and Siddharth Garg (2024). “LipSim: A Provably Robust Perceptual Similarity Metric”. In: *The Twelfth International Conference on Learning Representations* (cit. on p. 16).

Gnassounou, Théo, Antoine Collas, Rémi Flamary, Karim Lounici, and Alexandre Gramfort (2024). “Multi-Source and Test-Time Domain Adaptation on Multivariate Signals using Spatio-Temporal Monge Alignment”. In: *arXiv* (cit. on pp. 60, 216).

Havens, Aaron J, Alexandre Araujo, Huan Zhang, and Bin Hu (2024). “Fine-grained Local Sensitivity Analysis of Standard Dot-Product Self-Attention”. In: *Proceedings of the 41st International Conference on Machine Learning* (cit. on p. 19).

Hu, Kai, Klas Leino, Zifan Wang, and Matt Fredrikson (2024). “A Recipe for Improved Certifiable Robustness”. In: *The Twelfth International Conference on Learning Representations* (cit. on pp. 28, 150, 151).

Prach, Benedikt, Florian Brau, Giuseppe Buttazzo, and Christoph H. Lampert (2024). “1-Lipschitz Layers Compared: Memory, Speed, and Certifiable Robustness”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 27, 28, 215).

Voráček, Václav (2024). “Treatment of Statistical Estimation Problems in Randomized Smoothing for Adversarial Robustness”. In: *arXiv preprint arXiv:2406.17830* (cit. on pp. 114, 117, 118).

Yashwanth, M., Gaurav Kumar Nayak, Harsh Rangwani, et al. (2024). “Minimizing Layerwise Activation Norm Improves Generalization in Federated Learning”. In: *Winter Conference on Applications of Computer Vision (WACV)* (cit. on p. 43).

Boissin, Thibault, François Mamalet, Thibaut Fel, Anne-Marie Picard, Thibault Massena, et al. (2025). “An Adaptive Orthogonal Convolution Scheme for Efficient and Flexible CNN Architectures”. In: (cit. on pp. 24, 77).

Gonon, Antoine, Nicolas Brisebarre, Elisa Riccietti, and Rémi Gribonval (2025). “A Rescaling-Invariant Lipschitz Bound Based on Path-Metrics for Modern ReLU Network Parameterizations”. In: *Proceedings of the 42nd International Conference on Machine Learning (ICML)* (cit. on p. 148).

List of Figures

List of Tables

RÉSUMÉ

L'apprentissage profond a connu des succès remarquables dans de nombreux domaines, mais ses modèles restent fragiles et instables : de petites perturbations des données d'entrée peuvent entraîner de fortes variations des prédictions, et l'optimisation est souvent affectée par des paysages de perte trop abrupts. Cette thèse aborde ces problèmes à travers le prisme unificateur de l'analyse de sensibilité, qui étudie la réponse des réseaux de neurones aux perturbations des entrées et des paramètres.

Nous explorons les réseaux Lipschitziens comme moyen de contrôler la sensibilité aux perturbations des entrées, améliorant ainsi la généralisation, la robustesse face aux attaques adversariales et la stabilité de l'entraînement. En complément, nous introduisons des techniques de régularisation basées sur la courbure de la fonction de perte, afin de favoriser des paysages plus lisses et de réduire la sensibilité aux paramètres. Le lissage aléatoire est également étudié comme méthode probabiliste pour renforcer la robustesse au voisinage des frontières de décision.

En combinant ces approches, nous proposons un cadre unifié reliant la continuité Lipschitzienne, le lissage aléatoire et la régularisation par courbure pour traiter les défis fondamentaux de la stabilité. La thèse apporte à la fois des analyses théoriques et des méthodes pratiques, incluant le calcul efficace de normes spectrales, de nouvelles couches contraignant la constante de Lipschitz, et des procédures de certification améliorées.

MOTS CLÉS

Stabilité, Réseaux Lipschitz, Robustesse, Norme spectrale, Lissage aléatoire, Régularisation de Hessienne, Apprentissage profond

ABSTRACT

Deep learning has achieved remarkable success across a wide range of tasks, but its models often suffer from instability and vulnerability: small changes to the input may drastically affect predictions, while optimization can be hindered by sharp loss landscapes. This thesis addresses these issues through the unifying perspective of sensitivity analysis, which examines how neural networks respond to perturbations at both the input and parameter levels.

We study Lipschitz networks as a principled way to constrain sensitivity to input perturbations, thereby improving generalization, adversarial robustness, and training stability. To complement this architectural approach, we introduce regularization techniques based on the curvature of the loss function, promoting smoother optimization landscapes and reducing sensitivity to parameter variations. Randomized smoothing is also explored as a probabilistic method for enhancing robustness at decision boundaries.

By combining these perspectives, we develop a unified framework where Lipschitz continuity, randomized smoothing, and curvature regularization interact to address fundamental challenges in stability. The thesis contributes both theoretical analysis and practical methodologies, including efficient spectral norm computation, novel Lipschitz-constrained layers, and improved certification procedures.

KEYWORDS

Stability, Lipschitz networks, Robustness, Spectral norm, Randomized smoothing, Hessian regularization, Deep learning