Sant'Anna

School of Advanced Studies – Pisa

Academic Year 2022/2023

Phd Course

Emerging Digital Technologies

Methods for Certifiable Robustness of Deep Neural Networks

Author

Fabio Brau

Supervisor

Prof. Giorgio Buttazzo

ISBN: XXXXXXXXXXXX

Abstract

This Ph.D. thesis delves into the critical issue of enhancing formal and certifiable guarantees for the robustness of deep neural networks against input perturbations. It explores novel methodologies across five comprehensive chapters. Firstly, the thesis contributes to ongoing efforts to improve the certifiable robustness of deep neural networks. It introduces the problem of online estimation of minimal adversarial perturbation (MAP), focusing on the problem from a geometrical perspective. Secondly, the thesis transitions into achieving certifiable robust models through Lipschitz bounded neural networks. Finally, in pursuit of the same goal, a novel family of classifiers (SDC) is proposed and compared with the related family of Lipschitz bounded models.

In detail, the thesis proposes two root-finding-based strategies for estimating the MAP and provides theoretical results on the goodness of the estimation. Such a theoretical finding can be leveraged for an online estimation of the robustness of a classifier in a given input close enough to the classification boundary. Indeed, the approximate MAP, obtained with the proposed approaches, results in being less computationally expensive than the one obtained with the state-of-the-art methods, enabling a fast estimation of the ε-robustness of a classifier for the sample. Furthermore, the goodness of the estimation is linked to a model-dependent value, named boundary proximity index $\mathcal{K}_f(\Omega_\delta)$, which encapsulates the regularity of the decision boundary. Subsequently, the thesis addresses the challenge of designing 1-Lipschitz neural networks, which represent a tangible and effective method for developing certifiably robust classifiers. Therefore, this work includes an extensive comparison of the current state-of-the-art methods for designing 1-Lipschitz DNNs. It goes further to offer practical suggestions and guidelines for the usage of 1-Lipschitz layers, enhancing their effectiveness for the deployment of these models in safety-critical systems. Finally, a new family of classifiers, named Signed Distance Classifiers (SDCs), is discussed. A signed distance classifier provides as output not only the prediction of the class of a given input x but also the distance of x from the classification boundary. We provide a theoretical characterization of the SDCs and propose a tailored network architecture, named UGNN, which, to the best of our knowledge, represents the first practical approximation of an SDC.

In conclusion, the thesis provides a comprehensive overview of three main directions for achieving certifiable robustness for deep neural networks, by representing a modest yet significant stride towards the application of deep neural networks in safety-critical systems.

Contents

List of Figures

Chapter 1

Introduction

ADVERSARIAL EXAMPLES (AEs) are a decade-old phenomenon that occurs in the machine learning models, first highlighted by Biggio *et al.* [1] under the name of *evasion attacks*, and independently discovered by Szegedy *et al.* [2] in deep neural networks. In the field of image classification, adversarial examples are meticulously crafted inputs that, while visually indistinguishable to humans from unaltered images, lead to erroneous classifications by the models. This definition is extendable to a broader spectrum of machine learning detection or classification models. Here, an adversarial example is a maliciously crafted input with high similarity to natural uncorrupted samples, yet remains difficult to detect, ultimately resulting in the failure of the machine learning model to perform accurately. This dual characteristic underscores the critical challenge adversarial examples pose to the integrity and reliability of machine learning systems.

This phenomenon presents two main interpretations within the field of academic discourse. On one hand, adversarial examples (AEs) are recognized as a significant *security threat*, which imposes considerable limitations on the application of machine learning in cyber-physical systems, raising profound security implications in critical areas such as spam and malware detection, object detection, and semantic segmentation, [3], [4]. On the other hand, despite the large amount of research, AEs are acknowledged as a partially understood phenomenon, shedding light on the *black-box* nature of deep neural networks. A continuously growing literature highlights the challenges related to the instability and robustness of such models, by also pointing the finger at the high Lipschitz constant of the — standard — deep neural networks as a direct cause of sensitivity to input perturbation. This thesis will delve into the second interpretation, seeking to enhance our comprehension of AEs and their role and impact on the functioning of neural networks.

The scientific community initially focused on establishing the existence of AEs, and on the development of algorithms to find them, aiming at developing a comprehensive understanding of the phenomenon and devising effective countermeasures to mitigate the sensitivity of DNNs to these perturbations. For this purpose, we only focus on the attacks in a *white-box* setting, where the attacker has full knowledge of the model's architecture and parameters. In particular, the *Fast Gradient Sign (FGSM)* method, proposed in [5], has been used to produce a first attempt of *adversarial training*, a technique that aims to improve the robustness of a model by augmenting the training set with adversarial examples. Furthermore, the *DeepFool (DF)* [6] method, is proposed instead as an empirical measurement of the *robustness* of a neural model; meant as an estimated-measure of the sensitivity to the perturbation with a given magnitude that produce a failure of the classification. Finally, more recent algorithms, such as the methods proposed in [7]–[11], were designed to find the minimal perturbation that produces a failure of the classification in a given input.

For the sake of providing certifiable guarantees on the safety and reliability of the prediction in a given sample, formal countermeasures became the focal point of the scientific community's efforts. For this purpose, *Verification Methods* aim to formally check, for a given input and a threshold ε, the sensitivity of the prediction against effective adversarial perturbation of magnitude that does not exceed the threshold ε. This effectively provides a measure of trustworthiness for any given input. Theoretically, such strategies naturally lead towards a certifiable-safe model, obtained by adopting a detection-rejection-strategy of unsafe inputs on top of the original one. In practice, however, the computational complexity of these methods is prohibitive, see [12], [13], hence the resulting models are not suitable for real-world applications. This aspect mainly motivates the results discussed in the Chapter 2, in which a lighter version of the analysis provided in [14] is proposed. The final goal of the work is to provide an approximated solution to the verification problem, presented as the problem of finding the minimal adversarial perturbation. The analysis brought to an interesting factor named *Boundary Proximity Index (BPI)* that, once calculated, can provide formal guarantees on the two methods for the approximated solution of the verification problem. Unfortunately, such a coefficient can not be explicitly calculated, hence the experimental part aims at confirming the theoretical findings, and at providing a way to estimate the BPI in a real-world scenario.

Nevertheless, as one might expect, a naive approach to robust classification through a detection-rejection strategy, as discussed in [15], is insufficient for ensuring a robust model. Indeed, without additional countermeasures, a standard neural network exhibits a notably low empirical robustness, as shown in the paper mentioned above. Such networks typically position input samples close to decision boundaries, making them highly susceptible to adversarial perturbations. This leads to a high likelihood of rejecting input samples, which in turn results in reduced accuracy of the classifier. Adversarial training strategies, [16], can alleviate this problem by presenting adversarial examples during the training stage of the model, improving the robustness by increasing the margin between the input samples and the decision boundaries. Nevertheless, even though increasing the empirical robustness, such strategies do not provide certifiable guarantees at inference time, and hence, without external supports (e.g. given by verification and detection strategies), they are still not enough adequate for the reliable deployment of a machine learning model in cyber-physical systems.

Therefore, the pursuit of an efficient and accurate robust model has prompted the scientific community to explore various approaches. Beginning with the seminal work of [2], it was established that the Lipschitz constant of a neural classifier is closely linked to the model's instability, as evidenced by the existence of adversarial examples. Consequently, one way to address the instability problem can involve constraining the Lipschitz constant of a model by limiting the spectral radius of its weights, which has proven to be an effective countermeasure, but not without challenges and concerns. Indeed, training a model with a bounded Lipschitz constant requires a substantial effort, and results in a robust accuracy that may not rival that of a more straightforward model.

Lipschitz bounded neural networks can provide certifiable guarantees on the sensitivity of the model in a given input x. In detail, for such models, the difference between the two largest output components of the DNN directly provides a lower-bound, in Euclidean norm, of the minimal adversarial perturbation capable of fooling the model in x. The Chapter 3 provides a comparison of various methods for obtaining 1-Lipschitz neural networks known in the literature, based on the results presented in the paper [17], which is currently under review. Indeed, although different methods for achieving 1-Lipschitz DNNs have been proposed in the literature, understanding the performance under different metrics (e.g., training time, memory usage, accuracy, certifiable robustness) is not straightforward but yet extremely useful, since they have different impacts on different applications. For this reason, the chapter provides a thorough theoretical and empirical comparison between methods by evaluating them in terms of memory usage, speed, and certifiable robust accuracy. In conclusion, some guidelines and recommendations are given, to support the user in selecting the methods that work best depending on the available resources.

As anticipated above, 1-Lipschitz DNNs allow *online verification through a single forward of the model*. Nevertheless, only a lower bound of the smallest adversarial perturbation can be deduced, possibly resulting in a large number of *false postive* rejections, *i.e.* samples for which the lower bound is lower than a certain safe-threshold ε, but practically robust against perturbation of magnitude lower than ε. For this reason, in the Chapter 4, a novel classification model named *signed distance classifier* is proposed, based on the results discussed in the paper [18].

The (SDCs), from a theoretical perspective, directly output the exact distance of a given input from the classification boundary, rather than a probability score (e.g., SoftMax), and, like any other 1-Lipschitz model, represents a family of robust-by-design classifiers. As proved in the dedicated section, a signed distance classifier is characterized (in a region close to the decision boundary) by the property of having a gradient with a constant Euclidean norm equal to 1. To practically address such a theoretical requirement of an SDC, a network architecture named *Unitary-Gradient Neural Network* is presented, that leverages the 1-Lipschitz orthogonal layers in order to guarantee the unitary-gradient property. The chapter is comprehensive of experimental results that show that the proposed architecture approximates a signed distance classifier, hence allowing an online certifiable classification of any input at the cost of a single inference. As shown in the experimental part, even though the estimation of the minimal adversarial perturbation is particularly accurate, the resulting classification accuracy does not outperform the accuracy of the strictly related Lipschitz-bounded models.

1.1 Structure of the thesis

The structure of the thesis is summarized as follows

- The **Chapter 1** serves as an introduction to the main scope of the thesis, effectively guiding the reader through the motivations behind each chapter. It lays the groundwork for understanding the broader context and objectives of the thesis. The aim is to provide an aerial view of the three key works proposed in the thesis, illustrating how they are interconnected and contribute to the overarching theme. This approach helps in establishing a cohesive narrative, connecting individual research efforts to the larger goal of advancing the field of machine learning with a focus on adversarial examples and model robustness.

- **Chapter 2** contains a reformulated version of the work proposed in the paper [14], where the Theorem's proof has been refined, and a greater focus on the property of the model is given by explicitly defining the boundary proximity index.

- **Chapter 3** aggregates the results proposed in the paper currently under review, [17], and integrates them with the corresponding appendix to form a unified and coherent narrative flow. It is important to clarify that the paper is a ch-authored work. Specifically, my contributions mostly involve the comparisons of certifiable robustness including the random-search needed for the hyper-parameters tuning, while Bernd's contributions mostly involve the theoretical estimation of the algorithmic complexity, as well as the experimental estimation of the memory usage and timing.

- **Chapter 4** proposes the signed distance classifier discussed in [18], including a datailed proof of the characterization theorem.

- The **Chapter 5** presents the key takeaways and potential future research directions, effectively concluding the elaborate.

Chapter 2

Minimal Adversarial Perturbation

In the last decade, Deep Neural Networks (DNNs) achieved impressive performance on computer vision applications, such as image classification [19] and object detection [20]. Despite their excellent results, all those models are liable to adversarial attacks, defined as input perturbations intentionally designed to be undetectable to humans but causing the model to make a wrong output [1], [2]. Extensive studies have been conducted to improve these attacks through effective techniques that minimize the distance from the original input to make the resulting adversarial input imperceptible to humans.

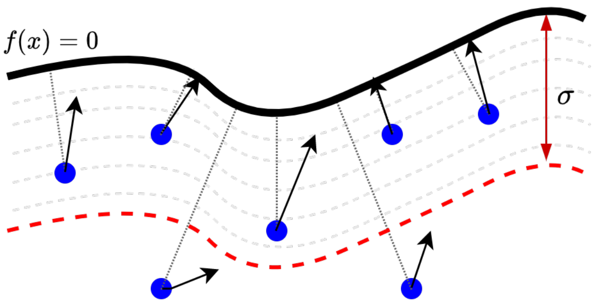


Figure 2.1: Illustration of the addressed problem. The blue points are DNN inputs, while the black line $f(x) = 0$ is the classification boundary that distinguishes points belonging to the class -1 ($f(x) < 0$) and class 1 ($f(x) > 0$). The dotted line starting from each point is the unknown optimal perturbation, which is orthogonal to the classification boundary. The black arrows represent the gradient directions. Observe that the gradients computed on the points whose distance from the boundary is closer than σ provide a good approximation to the minimal adversarial distance.

Finding the closest adversarial example or in other terms, the *Minimal Adversarial Perturbation (MAP)* capable of fooling the model, is a notoriously hard problem, because it involves the solution of a non-convex optimization problem with highly-irregular constraints, due to the intrinsic nature of DNNs [2], [6]–[8]. At present, this is still a hot research topic since allows retrieving useful information on the robustness of the model under adversarial attacks. Almost all the powerful attacks presented in the literature (e.g., [2], [5]–[9], [16]) rely on the loss function gradient to build up optimization methods for crafting those perturbations. In a nutshell, their idea is moving toward the direction that mostly increases the loss function, thus increasing the probability of misclassification.

In light of these observations, our study concentrates on analyzing an approximate solution for the minimal adversarial perturbation problem. The goal is to devise a method that is as swift as possible, primarily relying on fewer forward and backward passes, supported by theoretical findings on the existence and the accuracy of a solution. To achieve this, we proposed two strategies, that consistently offer an upper bound of the exact solution, that rely on a few amount of inferences of the model; e.g., in one of the proposed methods, we reduce the backward passes to just one and limit the forward computations to fewer than 20. Due to the simplicity of these strategies, we have also focused on providing theoretical analysis. This analysis aims at supporting the methods in two ways: It demonstrates the existence of a solution in a region close to the decision boundary; it provides an estimation of the error committed depending on the first and the second derivative of the classifier f. Furthermore, empirical evaluations show that such a theoretical error estimation can be applied to a real-case scenario (classification through deep neural networks). In particular, we exploit it to estimate the robustness of a sample within a specific neighborhood with high confidence.

More specifically, Section 2.1 proves the existence of a tubular neighborhood with radius σ where the error between the approximate and the exact solution can be bounded. An illustration of such an estimation can be found in Figure 2.1, where x is the input vector, $f(x)$ is the classification function learned by the network, and the red line represents the radius σ. Such a radius depends on a coefficient named *boundary proximity index (BPI)* that measures the reachability of a solution only by following the gradient direction and that only depends on the first and second derivative of the classifier. Section 2.4 reports a set of experiments with two specific aims. The first aim is to compare the distance computed by the proposed approximated strategies with the solution of a costly method based on a global search. The second aim is to devise an empirical estimation of the radius σ (that can not be explicitly computed) and using it to estimate the robustness of the model in a given input with high confidence. In summary, this chapter

- Proposes two strategies based on a root-finding algorithm to approximate the solution of the minimal adversarial perturbation problem in a region close to the classification boundary.
- Provides an analytical estimation of the neighborhood in which the previous analysis holds by leveraging a novel coefficient that measures the regularity of the classifier.

2.1 Definitions and Theoretical discussions

In this section, we will analyze and provide an answer to the following questions.

Is the only gradient direction in a given sample capable of providing a good estimation of the minimal adversarial perturbation, i.e, the smallest perturbation the fools a classifier? How can the precision of such a approximation be bounded with respect to actual MAP, according to the regularity of the classifier?

Through this section, we show that, under certain assumptions, the solution of the minimal adversarial perturbation problem has a natural and intuitive estimation obtained by following the gradient direction. Furthermore, we show the relation between such an estimation with the first and the second derivative of the classifier $\nabla f(x)$ and $\nabla^2 f(x)$.

Henceforth, we consider a classifier modeled by a continuous function that accepts an input vector of length n and returns a discrete result ranging from 1 to C, where C denotes the total number of classes that can be identified. Specifically, consider $f : \mathbb{R}^n \to \mathbb{R}^C$ as a continuous function. For a given input x, the *predicted class* $\hat{k}(x)$ is determined as the index that corresponds to the highest component of $f(x)$. Formally, $\hat{k}(x)$ is defined so that $f_{\hat{k}(x)}(x) > f_k(x)$ for all $k \neq \hat{k}(x)$. If the highest value is not unique, that is, $f_{\hat{k}(x)}(x) = \max_{k \neq \hat{k}(x)} f_k(x)$, then we assign $\hat{k}(x) = 0$ to indicate an unreliable classification. Moreover, it is useful to define $R_j := \{x \in \mathbb{R}^n : \hat{k}(x) = j\}$ as the region in the input space corresponding to class j, and $\mathcal{B}_j$ as the classification boundary for class j, or the edge of R_j. Let x be a sample classified with label l. The problem of finding the *minimal adversarial perturbation* δ^*, such that $x + \delta^*$ is the *closest adversarial example* to x, can be formally stated with the following minimum problem

$$\begin{aligned} d(x,l) = \inf_{\delta \in \mathbb{R}^n} \quad & \|\delta\| \\ \text{s.t} \quad & \hat{k}(x+\delta) \neq l, \end{aligned} \tag{MAP}$$

where $\|\cdot\|$ is the euclidean norm, and the scalar value $d(x,l)$ represents the distance between x and the closest adversarial example $x+\delta^*$, or, equivalently, the distance of x from the classification boundary $\mathcal{B}_l$.

Constrained minimal adversarial perturbation

Note that the formulation given by Equation (MAP) is merely theoretical since does not take into account two important constraints: the *box-constraint* and the *integer-constraint*. When the input of the classifier is constituted by images, the box-constraint ensures that the solution δ is a perturbation such that $0 \leq x+\delta \leq 1$. This particularly belongs to the fact that typical inputs of classifiers are images, for which the pixel values are normalized to be in the range $[0, 1]$. For the same reason — i.e, when images are the input of the classifier — the integer-constraint ensures that each pixel $(x + \delta)_i$ is discretized into Q color-levels (e.g., $Q = 256$), that is, $Q \cdot (x_i + \delta_i) \in [0, Q - 1] \cap \mathbb{N}$. For the sake of completeness, we report the adversarial perturbation problem with constraints (C-MAP) as,

$$\begin{aligned} d(x,l) = \inf_{\delta \in \mathbb{R}^n} \quad & \|\delta\| \\ \text{s.t} \quad & \hat{k}(x+\delta) \neq l, \\ & 0 \leq x+\delta \leq 1 \\ & Q \cdot (x+\delta) \in ([0, Q-1] \cap \mathbb{N})^n \end{aligned} . \tag{C-MAP}$$

Nevertheless, note that this section only focuses on the unconstrained formulation, as similarly done in [6], since the analytical instruments are more compliant with this formulation. However, note that this aspect does not reduce generality, since the solution of MAP provides a lower bound of the constrained problem C-MAP. Therefore, to reduce clutter, unless differently specified, the domain of the perturbation δ is equal to $\mathbb{R}^n$.

2.1.1 The case of a binary classifier

To enhance understanding, we initially focused on the binary classification case. A detailed discussion on the extension of the results to the general multiclass scenario is presented in Section 2.3.4. In contrast to a multiclass classifier, a *binary classifier* is represented as a scalar function $f : \mathbb{R}^n \to \mathbb{R}$, from which the classification is deduced looking at the sign of its output, that is, for each $x \in \mathbb{R}^n$, $\hat{k}(x) = \operatorname{sgn}(f(x))$. It is also important to note that for a binary classifier, the classification boundary is independent of the class. Thus, without loss of generality, we can define the boundary as the set of zeros of the function f, that is $\mathcal{B} = \{x \in \mathbb{R}^n : f(x) = 0\}$.

Furthermore, observe the MAP problem can be reformulated differently. Indeed, given x of class l, the minimum distance from the region other than l coincides with the distance from the decision boundary. In other terms, the inequality constraint can be substituted by an equality constraint $f(x + \delta) = 0$. This fact can be proved by observing that, if δ^* is a solution of MAP, and if $\operatorname{sgn}(f(x)) \neq \operatorname{sgn}(f(x + \delta^*)) \neq 0$, then, due to the continuity of f, there exists $0 < t < 1$ such that $f(x + t\delta^*) = 0$, and hence $\|t\delta^*\| < \|\delta^*\|$, which is a contradiction. Based on this observation, we can replace the original problem with the following minimization problem with an equality constraint

$$\begin{aligned} d(x,l) = d(x) = \inf_{\delta} \quad & \|\delta\| \\ \text{s.t} \quad & f(x+\delta) = 0, \end{aligned} \tag{B-MAP}$$

equivalent to a minimum distance problem from set $\mathcal{B}$. Finally, since the aim of the section is to analyze how the gradient direction is effective in estimating the solution of MAP, we must define the following minimal-root problem

$$\begin{aligned} t(x) = \inf_{t \in \mathbb{R}_+} \quad & t \\ \text{s.t} \quad & f(x + t\nu(x)) = 0 \end{aligned} \tag{B-RMP}$$

where $\nu = -\operatorname{sgn}(f(x)) \frac{\nabla f(x)}{\|\nabla f(x)\|}$.

Before going deeper into the mathematical details, it is worth observing that the gradient $\nabla f(p)$ is orthogonal to the boundary $\mathcal{B}$ for each $p \in \mathcal{B}$, and that, if x is close to the boundary, then $\nabla f(x) \approx \nabla f(p^*)$ (where $p^* = x + \delta^*$) provides the fastest direction to reach the boundary. Hence, it is natural to observe that is reasonable to approximate B-MAP with the minimal root problem B-RMP. The goodness of such an estimation can be summarized in the Section 2.1.1, where we show that in a neighborhood of the decision boundary, the precision of the estimation is strongly related to a certain factor that depends on the first and second derivative of the classifier f, as informally summarized in the following Theorem.

Theorem (Informal MAP estimation)**.** *Close enough to the decision boundary, the solution of B-MAP can be estimated by the solution of B-RMP through the following inequality*

$$\frac{1}{\rho} t(x) \leq d(x) \leq t(x),$$

where the ρ is a constant factor, and the neighborhood for which the inequality holds depends on the following ratio

$$\frac{\inf_{x\in\Omega}\|\nabla f(x)\|}{\sup_{x\in\overline{\Omega}}\|\nabla^2 f(x)\|}. \tag{2.1}$$

The set Ω is a large neighborhood of $\mathcal{B}$ with compact closure.

The proof of the estimation theorem leverages the regularity of the classifier f and the proximity of the sample to the decision boundary. For this reason, let us introduce the definition of *tubular neighborhood* of the decision boundary.

Definition 1 (Tubular neighborhood)**.** Given a positive number $\sigma > 0$, the tubular neighborhood of the boundary Ω_σ of radius σ is the set of the samples whose distance from the classification boundary $\mathcal{B}$ is less than σ. In formulas,

$$\Omega_\sigma := \{x \in \mathbb{R}^n \,:\, d(x) < \sigma\}.$$

Henceforth, we assume that the function f satisfies the following properties.

Assumption A. The function f is of class $C^2(\mathbb{R}^n)$, i.e., is derivable twice and the second derivative is continuous.

Assumption B. The function f is strictly positive outside some compact.

Assumption C. The gradient ∇f is not zero in $\mathcal{B}$ (i.e., 0 is a regular value of f).

Although the three Assumptions A, B,C are not valid in general, they are not restrictive for a common neural-classifier. Further details on this aspect will be deepened in Section 2.1.2. Assumption B ensures that $\mathcal{B}$ is a *compact set*, and hence that the minimum distance problem formulated in B-MAP admits a solution for each $x \in \mathbb{R}^n$. Nevertheless, there is no guarantee that for each $x \in \mathbb{R}^n$ there exists a unique closest point in $\mathcal{B}$. Assumption C ensures that there exists an open neighborhood Ω_δ of the boundary such that $\nabla f_{\restriction\overline{\Omega_\delta}} \neq 0$. Observe also that for any $\delta' \leq \delta$ the gradient never zeros in $\Omega_{\delta'}$. As anticipated above, the regularity of the classifier has a big impact on the estimation theorem, thus we propose the following measure of regularity.

Definition 2 (Boundary Proximity Index)**.** For a given binary classifier f the *Boundary Proximity Index* (BPI), on a neighborhood $\Omega \supset \mathcal{B}$ with a compact closure $\overline{\Omega}$, expresses a measure of the reachability of the decision boundary considering

$$\mathcal{K}_f(\Omega) = \frac{\inf_{x\in\Omega}\|\nabla f(x)\|}{\|\nabla^2 f\|_{\overline{\Omega}}}, \tag{BPI}$$

where the shortcut $\|\cdot\|_{\overline{\Omega}}$ is the maximum operatorial norm on the compact, i.e.,

$$\|\nabla^2 f\|_{\overline{\Omega}} = \max_{x\in\overline{\Omega}}\|\nabla^2 f(x)\|.$$

We anticipate that the BPI is a global measure of the reachability of the decision boundary from any point in the open set Ω. Observe that such an index is not defined for a linear classifier, that indeed has a not compact boundary since does not satisfy Assumption B. However, if we though a linear classifier

$g(x) = w^T x + b$ as the limit of $g_t(x) = \frac{1}{t}\|x + \frac{1}{2}t\,w\|^2 + b$ for $t \to \infty$, it is natural to estend BPI, with $\mathcal{K}_g(x) = +\infty$.

The proof of the main theorem will be decomposed into three main steps, each one summarized in the dedicated lemma to facilitate readability.

1. A first Lemma, provides a bound of the angle between the direction given by the gradient w.r.t. the optimal direction. The lemma provides a radius σ_1 (depending on the BPI) for which the bound holds;
2. A second Lemma, bounds the portion of the surface that contains the optimal solution of B-MAP between two hyperplanes. Another radius σ_2 is given, depending on the same index;
3. Finally, the proof of the Theorem shows that an intersection between the hyperplanes and gradient direction exists. This fact directly gives an estimation of the solution of B-RMP, and thus an estimation of the optimal solution.

Lemma 1 (Angular Bound)**.** *Let $\Omega_\delta \supset \mathcal{B}$ be a tubular neighborhood such that $\nabla f_{\restriction \bar{\Omega}} \neq 0$. For any measure of similarity $\alpha \in (0,1)$, let $\sigma_1(\alpha) = \frac{1-\alpha}{2}\mathcal{K}_f(\Omega_\delta)$. Then for each $x \in \Omega_{\sigma_1(\alpha)}$, the following inequality holds*

$$\frac{\nabla f(x)^T \nabla f(\pi(x))}{\|\nabla f(x)\|\|\nabla f(\pi(x))\|} > \alpha, \tag{2.2}$$

where $\pi(x)$ is a solution of B-MAP.

Proof. Let $p \in \mathcal{B}$. By the definition of Ω_δ, the function $F_p(x) = \left\langle \frac{\nabla f(x)}{\|\nabla f(x)\|}, \frac{\nabla f(p)}{\|\nabla f(p)\|} \right\rangle$ is differentiable in the neighborhood of p of radius δ, i.e., $F_p \in C^1(B(p,\delta))$, and, hence, satisfies the hypothesis of Taylor Theorem in several variables, [21]. In detail

$$\nabla F_p(x) = \left(\frac{\nabla^2 f(x)}{\|\nabla f(x)\|} - \frac{\nabla f(x) \nabla f(x)^T \nabla^2 f(x)}{\|\nabla f(x)\|^3} \right) \frac{\nabla f(p)}{\|\nabla f(p)\|}$$

is a continuous vector field in the ball of radius δ, and so for each $x \in B(p,\delta)$ there exists $\xi \in B(p,\delta)$ such that

$$F_p(x) = 1 + (x-p)^T \nabla F_p(\xi). \tag{2.3}$$

By rearranging the terms of Equation (2.3) and considering the supremum on the right side, we can deduce the following chain of inequalities

$$\begin{aligned}
|1 - F_p(x)| &\leq \max_{\xi \in \bar{\Omega}} |(x-p)^T \nabla F_p(\xi)| \\
&\leq \max_{\xi \in \bar{\Omega}} \|\nabla F_p(\xi)\| \cdot \|p - x\| \\
&\leq \max_{\xi \in \bar{\Omega}} \left\| \frac{\nabla^2 f(\xi)}{\|\nabla f(\xi)\|} - \frac{\nabla f(\xi) \nabla f(\xi)^T \nabla^2 f(\xi)}{\|\nabla f(\xi)\|^3} \right\| \cdot \|p - x\| \qquad (2.4) \\
&\leq \max_{\xi \in \bar{\Omega}} \left\| Id - \frac{\nabla f(\xi)}{\|\nabla f(\xi)\|} \frac{\nabla f(\xi)^T}{\|\nabla f(\xi)\|} \right\| \cdot \left\| \frac{\nabla^2 f(\xi)}{\|\nabla f(\xi)\|} \right\| \cdot \|p - x\|
\end{aligned}$$

where, for any matrix $A \in \mathbb{R}^{n\times n}$, the notation $\|A\|$ represents the operator-norm inducted by the Euclidean norm. Observing that for each $\|v\| = 1$, $\|Id - vv^T\| \leq 1 + \|vv^T\| \leq 2$, the first term can be simplified and so we can reduce the last inequality to the following

$$|1 - F_p(x)| \leq \frac{2\|\nabla^2 f\|_{\overline{\Omega}}}{\inf_{\xi\in\Omega}\|\nabla f(\xi)\|} \cdot \|p - x\| = \frac{2\|p - x\|}{\mathcal{K}_f(\Omega)}. \tag{2.5}$$

By rearranging the terms, we can finally deduce that

$$F_p(x) \geq 1 - \frac{2\|x - p\|}{\mathcal{K}_f(\Omega)},$$

from which we can deduce that the condition $1-\frac{2\|x-p\|}{\mathcal{K}_f(\Omega)} > \alpha$ is a sufficient condition to $F_p > \alpha$ for each $x \in B(p, \delta)$. Rearranging the terms, we deduce that

$$\|p - x\| \leq \frac{1-\alpha}{2}\mathcal{K}_f(\Omega_\delta) = \sigma_1(\alpha), \quad \Rightarrow \quad F_p(x) > \alpha$$

Because $\sigma_1(\alpha)$ is an uniform estimation for each p, then we deduce the thesis for all the $x \in \Omega_\delta$ and $p = \pi(x)$. □

Intuitively, a small portion of the boundary can be enclosed between two affine parallel hyperplanes. This is the aim of the following lemma.

Lemma 2 (Hyperplanes Bound)**.** *Let $\Omega_\delta \supset \mathcal{B}$ be a tubular neighborhood such that $\nabla f_{\restriction\overline{\Omega}} \neq 0$. For each thickness factor $\beta \in (0, 1)$, let $\sigma_2(\beta) = 2\beta\mathcal{K}_f(\Omega_\delta)$. Then, for each $p \in \mathcal{B}$, the open set*

$$\Gamma_r(p) := \left\{p + v \,:\, |v^T\nabla f(p)| < \beta r\|\nabla f(p)\|,\, v \in \mathbb{R}^n\right\}$$

contains $\mathcal{B} \cap B(p, r)$ for any $r < \sigma_2(\beta)$.

Proof. Let $p \in \mathcal{B}$ and $q \in \mathcal{B} \cap B(p, r)$ where $r < \sigma_2(\beta)$. By applying the Taylor Theorem [21] to f centered in p, we deduce that there exists $\xi \in B(p, r)$ such that

$$0 = (p - q)^T\nabla f(p) + \frac{1}{2}(p - q)^T\nabla^2 f(\xi)(p - q) \tag{2.6}$$

from which, by considering the maximum over ξ on the right side, we can deduce the following inequality

$$|(p - q)^T\nabla f(p)| < \frac{1}{2}\|p - q\|^2\|\nabla^2 f\|_{\overline{\Omega}_\delta}. \tag{2.7}$$

Considering the following chain of inequalities

$$\begin{aligned}
&\|p - q\| < 2\beta\,\mathcal{K}_f(\Omega_\delta)\\
\Rightarrow\quad &\|p - q\| < 2\beta\,\frac{\|\nabla f(p)\|}{\|\nabla^2 f\|_{\overline{\Omega}_\delta}}\\
\Rightarrow\quad &\frac{1}{2}\|p - q\|\|\nabla^2 f\|_{\overline{\Omega}_\delta} < \beta\,\|\nabla f(p)\|\\
\Rightarrow\quad &\frac{1}{2}\|p - q\|^2\|\nabla^2 f\|_{\overline{\Omega}_\delta} < \beta\,\|p - q\|\|\nabla f(p)\|
\end{aligned} \tag{2.8}$$

we deduce that

$$|(p-q)^T \nabla f(p)| < \beta \|p-q\| \|\nabla f(p)\| < \beta r \|\nabla f(p)\|. \tag{2.9}$$

This shows that $q = p + (q-p) \in \Gamma_r(p)$ for any $q \in B(p,r)$, where $r < \sigma_2(\beta)$, from which we deduce the thesis. □

The Lemma above shows that the boundary $\mathcal{B}$ can be locally bounded by the open set $\Gamma_r(p)$ for each point p and for each radius r not larger than $\sigma_2(\beta)$. Furthermore, the border $\mathcal{B}$ splits the set $B(p,r) \cap \Gamma_r(p)$ in a way that f keeps a constant sign in the two hyperplanes $R_\pm := \{p + v : v^T \nabla f(p) = \pm \beta r \|\nabla f(p)\|, \, v \in \mathbb{R}^n\}$ that are included in the frontier of $\Gamma_r(p)$. This statement is condensed in the following observation, deduced as a direct consequence of Lemma 2.

Observation 1. Let $\beta \in (0,1)$ and $\sigma_2(\beta)$ of Lemma 2. Let $x \in \Omega_{\sigma_2(\beta)}$, $p = \pi(x)$ a solution of B-MAP, and let $r = \|x - p\|$. The hyperplane

$$R := \{p + v : v^T \nabla f(p) = -\mathrm{sgn}(f(x)) \beta r \|\nabla f(p)\|, \, v \in \mathbb{R}^n\}$$

is such that

$$\forall y \in R \cap B(p,r), \quad \mathrm{sgn}(f(y)) = -\mathrm{sgn}(f(x)). \tag{2.10}$$

Proof. Without loss of generality, we can initially assume $f(x) < 0$. The proof can be decomposed in two steps:

(i) Prove that $p_1 := p + r\beta \frac{\nabla f(p)}{\|\nabla f(p)\|} \in R$ and $f(p_1) > 0$;

(ii) Prove that if $y \in R \cap B(p,r)$, then $\mathrm{sgn} f(p_1) = \mathrm{sgn} f(y)$.

To prove the first statement, let $p_t := p + t\beta r \frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $t \in [0,1]$ be the segment going from p to p_1. Since f is differentiable in p, then

$$f(p_t) = r\, t\, \nabla f(p)^T \left(\frac{\nabla f(p)}{\|\nabla f(p)\|} \right) + o(p_t),$$

and because $o(p_t)/t \to 0$, we can deduce that $\mathrm{sgn}(f(p_t)) = \mathrm{sgn}(r\|\nabla f(p)\|) = 1$ for small t. Let us now prove by contradiction that f is positive in p_1. If $f(p_1) \leq 0$, then there exist $\tau^* \leq 1$ such that $f(p_{\tau^*}) = 0$. Hence, $\|p - p_{\tau^*}\| = |\tau^* r \beta|$, from which $p_{\tau^*} \in B(p, \tau^* r)$. Let us consider the smaller radius $r^* = \tau^* r$ and observe that $p_{\tau^*} \notin \Gamma_{r^*}(p)$. In fact, $\tau^* \beta r \frac{\nabla f(p)^T}{\|\nabla f(p)\|} \nabla f(p) = \beta r^* \|\nabla f(p)\|$ shows that p_{τ^*} lays on the topological border of the set $\Gamma_{r^*}(p)$. This brings to a contradiction for Lemma 2 being $p_{\tau^*} \in \mathcal{B} \setminus \Gamma_{r^*}(p)$. Furthermore, if $y \in B(p,r) \cap R$, the second statement can be proved by contradiction observing that, if $f(y) \leq 0$, then there exists $z \in R \cap B(p,r)$ for which $f(z) = 0$, and thus that $z \in \mathcal{B}$ and $z \notin \Gamma_r(p) \cap B(p,r)$, which brings to a contradiction by Lemma 2. In conclusion, the case $f(x) > 0$ can be deduced by considering the segment $p_t := p - t\beta r \frac{\nabla f(p)}{\|\nabla f(p)\|}$, to prove that $f(p_1) < 0$. □

The following observation ensures that the gradient $\nabla f(\pi(x))$ in a solution $\pi(x)$ of B-MAP in x, coincides with the optimal direction. This is a widely known result that can be proved by applying the Lagrangian Theorem. The details of the proof have been reported, expert readers can skip directly to the main theorem.

Observation 2. Let $x \in \mathbb{R}^n$, and let $\pi(x)$ a solution of B-MAP. Then, the optimal direction $\pi(x) - x$ is parallel to $\nabla f(\pi(x))$. In formulas,

$$\frac{x - \pi(x)}{\|x - \pi(x)\|} = \operatorname{sgn}(f(x)) \frac{\nabla f(\pi(x))}{\|\nabla f(\pi(x))\|}. \tag{2.11}$$

Proof. By construction, $\pi(x)$ is the solution of the minimum problem B-MAP. Then, by the *Lagrangian Necessary Condition Theorem* (see [22, p. 278]) under the Assumption C, there exists $\lambda^* \in \mathbb{R}$ such that $\nabla \mathcal{L}\left(\pi(x), \lambda^*\right) = 0$, where $\mathcal{L}(p, \lambda) = \|x - p\| + \lambda f(p)$ is the lagrangian of the minimum problem. Observe that $\nabla \mathcal{L}\left(\pi(x), \lambda^*\right) = 0$ implies that

$$\lambda^* \nabla f(\pi(x)) = \frac{x - \pi(x)}{\|x - \pi(x)\|}. \tag{2.12}$$

From this identity, considering the Euclidean norm at each side, we deduce that $|\lambda^*| = \frac{1}{\|\nabla f(\pi(x))\|}$. It remains to prove that $\operatorname{sgn}(\lambda^*) = \operatorname{sgn}(f(x))$. To prove this statement, we proceed in three steps: (i) we prove that the segment p_t that connects x to $\pi(x)$ is such that $\operatorname{sgn}(f(p_t)) = \operatorname{sgn}(f(x))$ for $t > 0$; (ii) we show that for $t \approx 0$, the sign of $\operatorname{sgn}(f(p_t))$ is equal to the sign of $\nabla f(\pi(x))^T (x - \pi(x))$; (iii) by leveraging identity Equation (2.12), we show that the sign of λ^* is equal to the sign of $\nabla f(\pi(x))^T (x - \pi(x))$. Let $p_t := \pi(x) + t(x - \pi(x))$ where $t \in [0, 1]$. Observe that $\operatorname{sgn}(f(x)) = \operatorname{sgn}(f(p_t))$ for each $t \in (0, 1]$. In fact, by contradiction, if there exists τ with $\operatorname{sgn}(f(x)) \neq \operatorname{sgn}(f(p_\tau))$, then, by the *Intermediate Zero Theorem* applied to function f, it would exist a $\tau_* \in (0, 1)$ such that $f(p_{\tau_*}) = 0$. This would imply that

$$\|x - p_{\tau_*}\| = \|(1 - \tau_*)(x - \pi(x))\| < \|x - \pi(x)\|,$$

which is a contradiction because $\|x - p_{\tau_*}\| < d(x)$ but $\pi(x)$ solves B-MAP. Based on this fact, observe that, since f is differentiable in p_0, then

$$\begin{aligned} f(p_t) &= f(p_0) + \nabla f(p_0)^T (p_t - p_0) + o(p_t) \\ &= t \nabla f(\pi(x))^T (x - \pi(x)) + o(p_t), \end{aligned}$$

where $o(p_t)/t \to 0$ when $t \to 0$, from which we deduce that for small t, $\operatorname{sgn}(f(p_t)) = \operatorname{sgn}\left(\nabla f(\pi(x))^T (x - \pi(x))\right)$. In conclusion, multiplying each term of Equation (2.12) by $\nabla f(\pi(x))^T$, we deduce that the sign of the first term of the equivalence is equal to $\operatorname{sgn}(\lambda^*)$, which concludes the proof. □

Lemma 1 and Lemma 2 are linked by the following intuitive connection. In a geometrical sense, $d(x)$ represents the length of the shortest path needed to reach the boundary, which is obtained by moving from x along $-\nabla d(x)$. Similarly, let $t(x)$ be the length of the path (if there exists one) required to reach the boundary by following the direction $\nu(x) = -\operatorname{sgn}(f(x)) \frac{\nabla f(x)}{\|\nabla f(x)\|}$, in formulas $x + t(x)\nu(x) \in \mathcal{B}$. To ensure the existence of such a $t(x)$, we can leverage two conditions. If we admit that $\nu(x)$ is not similar to the optimal one (i.e., we assume a $\alpha \not\approx 1$ in Lemma 1), then the existence of $t(x)$ would only be guaranteed by an almost straight boundary $\mathcal{B}$, which requires a thickness factor close to zero, $\beta \approx 0$. Vice versa, if we admit a highly irregular boundary (i.e., $\beta \not\approx 0$), then the existence of $t(x)$ would only be guaranteed by a direction $\nu(x)$ close to the optimal one. This would require $\alpha \approx 1$.

This is the main idea of the following theorem, which, by balancing the two parameters α and β, ensures: (i) The existence of $t(x)$; and (ii) The estimation of $d(x)$ through $t(x)$ defined in Equation (2.20). A graphical idea of the proof is depicted in Figure 2.2.

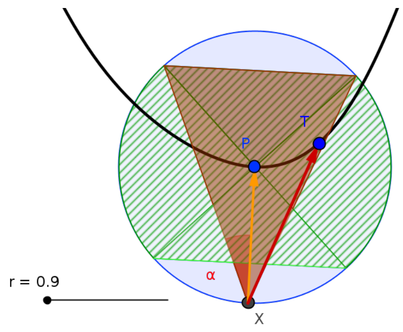


Figure 2.2: A graphical proof of Theorem 1. Lemma 1 ensures that $\nu(x)$ (in red) lays in the brown area. Lemma 2 ensures that in $B(p, r)$ the boundary belongs in the green area. In conclusion, there exists a solution T of B-RMP, i.e. an intersection between the boundary and the direction provided by the gradient.

Theorem 1 (MAP estimation)**.** *Let f be a binary classifier that satisfies the Assumptions A, B, C. Close enough to the decision boundary, the solution of B-MAP can be estimated by the solution of B-RMP through the following inequality. Formally, for each $\rho \in (\sqrt{2}, 2)$*

$$\forall x \in \Omega_{\sigma(\rho)}, \quad \frac{1}{\rho} t(x) \leq d(x) \leq t(x), \tag{2.13}$$

where the radius $\sigma(\rho) = \min\{\frac{1-\rho/2}{2}, \rho^2 - 2)\}\mathcal{K}_f(\Omega_\delta)$ and where Ω_δ is a — large — neighborhood of $\mathcal{B}$ in which $\nabla f(x) \neq 0$.

Proof. Let $\alpha = \frac{\rho}{2}$ and $\beta = \frac{1}{2}\rho^2 - 1$. Let $\sigma_1(\alpha)$ and $\sigma_2(\beta)$ be the radius deduced in Lemmas 1 and 2, respectively, and let $\sigma(\rho) = \min(\sigma_1(\alpha), \sigma_2(\beta))$. Let $p = \pi(x) \in \mathcal{B}$ a solution B-MAP, and let $\varphi(t) = x + t\frac{\nabla f(x)}{\|\nabla f(x)\|}$ be the straight line starting from x with direction $\nabla f(x)$. Without loss of generality, we can assume initially that $f(x) < 0$. The proof strategy consists of proving that the straight line $\varphi(t)$ intersects the hyperplane $R_+ := \left\{p + v \,:\, v^T \nabla f(p) = \beta d(x) \|\nabla f(p)\|,\, v \in \mathbb{R}^n\right\}$ of Observation 1 in some point $\varphi(t_*)$, and that $f(\varphi(t_*)) > 0$. Note that, this ensures the existence of some point $\varphi(t(x)) \in \mathcal{B}$, i.e., the existence of a solution of B-RMP. Observe that the intersection between the support of φ and R_+ is realized for

$$t_* = \frac{\|\nabla f(x)\|}{\nabla f(x)^T \nabla f(p)} \left(d(x)\beta \|\nabla f(p)\| - (x-p)^T \nabla f(p)\right). \tag{2.14}$$

Moreover, multiplying each term of Equation (2.12) in Observation 2 by $\nabla f(p)^T$, we deduce that $(x-p)^T \nabla f(p) = -r\|\nabla f(p)\|$, from which, by substituting in the second term of Equation (2.14), we deduce that

$$t_* = \frac{\|\nabla f(x)\| \|\nabla f(p)\|}{\nabla f(x)^T \nabla f(p)} (1+\beta)\, d(x). \tag{2.15}$$

Furthermore, we are in the hypothesis of Lemma 1, and hence $\frac{\|\nabla f(x)\|\|\nabla f(p)\|}{\nabla f(x)^T \nabla f(p)} < \frac{1}{\alpha}$. Hence, directly by Equation (2.15) we deduce the right-hand side of the following inequality

$$t_* < \frac{1+\beta}{\alpha} d(x) = \rho d(x). \tag{2.16}$$

In conclusion, by observing that $x = \varphi(0)$, if we prove that $f(\varphi(0)) < 0 < f(\varphi(t_*))$, we can deduce the existence of $t(x) < t_*$ such that $f(\varphi(t(x))) = 0$, which finally implies left inequality of Equation (2.13). The right side of Equation (2.13) is instead a direct consequence of the definition of $t(x)$ and $d(x)$. Finally, observe that the case $f(x) > 0$ can be deduced by applying the same procedure to the classifier $-f$. □

Note that the tightest estimation of $d(x, l)$ through $t(x, l)$, achieved with $\rho = \sqrt{2}$, results in a relative error lower than

$$\tilde{\varepsilon} = \frac{t(x) - d(x)}{d(x)} < \sqrt{2} - 1 \approx 0.414.$$

Finally, the aim of the following observation is to devise a notable value of ρ such that the inequality holds for a large radius σ.

Observation 3 (Exploiting the largest σ)**.** Let $\Omega = \Omega_\delta$ of Theorem 1, and let ρ^* obtained by solving $\frac{1}{2}(1 - \rho/2) = \rho^2 - 2$. Then $\sigma^* = \sigma(\rho^*)$ is a notevolus large radius for which Equation (2.13) holds.

Proof. Since, according to Theorem 1, $\sigma(\rho) = \min\{\frac{1-\rho/2}{2}, \rho^2 - 2)\}\mathcal{K}_f(\Omega_\delta)$, in order to exploit the inequality in the largest possible radius, we can search for the value of ρ such that

$$\max_{\sqrt{2}<\rho<2} \sigma(\rho).$$

The reader can check that the maximum is taken in ρ^*. □

In conclusion, observe that $\rho^* \approx 1.461$, and that $\sigma^* \approx 0.134 \cdot \mathcal{K}_f(\Omega_\delta)$.

2.1.2 Does the error estimations apply to DNN?

The results stated in the previous section have been proved under the three assumptions A,B,C that in general are not satisfied by a common feed-forward-neural network (especially with a ReLU activation function). In detail, for a feed-forward deep neural network, with a one-dimensional output, Assumption B is not verified by f. However, being the samples of our interest always in some closed limited set K, we can theoretically substitute f in the following proofs with another function $\tilde{f}$ that coincides with f in the compact K and that satisfies Assumption B. Similarly, Assumptions A and C are not valid in general, but we can assume that, in a practical domain, f is the quantization of a function $\tilde{f}$ that satisfies the conditions.

2.2 Existing methods

This section discusses the most related papers on the problem of finding the minimal adversarial perturbation, discussed in the Section 2.1. For the sake of clarity, we group them into different categories depending on the approaches followed for solving MAP.

2.2.1 Challenges in adversarial robustness

The literature related to adversarial robustness is quite vast. The problem of adversarial perturbations for DNNs was first introduced by [1] and independently by Szegedy et al. [2]. Since then, a large number of works followed for proposing more powerful attacks [6], [7], [9], [10], [16], detection mechanisms [23], [24], and defense strategies [25]–[27]. Most adversarial attacks use a gradient-based approach to craft adversarial perturbations. Although they generate impressive human undetectable adversarial examples, the reliability of the gradient direction is often taken for granted and no bound was ever provided on the error committed, with respect to the minimal theoretical perturbation.

Although many of the methods in the literature, provide an affordable empirical solution to the minimal perturbation problem, two main aspects have to be taken into account

Computational Cost Methods that accurately estimate the minimal adversarial perturbation necessitate a substantial number of forward and backward passes through the model. As will be discussed in subsequent sections, techniques utilizing either a penalty minimum problem or a projection mechanism demand between 20 to 20,000 forward and backward passes. Therefore, implementing these methods in an online scenario significantly increases overhead, reducing speed performance by at least a factor of 20 to 50 times. However, it's important to note that these works are not intended for online applications.

Theoretical Support Various methods in the literature approach the computation of the minimal adversarial perturbation problem in diverse ways. Some methods are based on well-researched formulations of constrained minimum problems, which, under specific assumptions about the model's properties, have theoretical foundations supporting the existence of a solution or convergence towards one. On the other hand, other methods, though empirically more effective and reliable, lack theoretical guarantees, making the existence of a solution and the convergence towards it challenging to prove. In summary, according to our research, there is no theoretical analysis that quantifies and limits the error incurred while approximating the minimal adversarial perturbation.

2.2.2 Penalty-based methods

A well-known technique to solve a minimum constrained problem is given by the *Iterative Penalty* (IP) [22]. For instance, [2] and [7] introduced a penalty term c and solved the following minimization problem:

$$\min_{\delta} \quad c \cdot \|\delta\| + \mathcal{L}(x+\delta, l) \tag{2.17}$$

where the hyper-parameter c is selected through a line search. The rationale of c is to balance the importance of the two terms in the cost function. The second term $\mathcal{L}$ represents a specific loss function that is positive in the region R_l and zero in $\cup_{j\neq l} R_j$. Carlini and Wagner analyzed different loss functions finding that $\mathcal{L}(x, l) = (f_l(x) - \max_{j\neq l} f_j(x))^+$ produces the most effective results, where $f^+ = \max\{0, f\}$.

It is worth noting that in both works of [2] and [24], a box constraint is added to achieve an adversarial perturbation that is feasible in the image domain. In particular, Szegedy et al. [2] exploited the Limited-memory Broyden–Fletcher–Goldfarb–Shanno Box-constrained (L-BFGS-B) optimizer [22] to directly solve the minimum problem with the box-constraint $0 \leq x + \delta \leq 1$. The Carlini and Wagner (CW) method instead, presented in [24], introduced a change-of-variable to reduce to the solution of an unconstrained problem. Although both the previous techniques allow crafting accurate perturbations, they turn out to be expensive in terms of memory usage and computational cost. Moreover, the optimization procedure must be repeated over multiple choices of the penalty c, causing a large number of forward and backward network passes, thus resulting in a slow convergence.

2.2.3 Toward the online computation of MAP

A key contribution towards less expensive solutions of MAP was given by the *Decoupling Direction Norm* method (DDN) presented by [9], where the authors avoid searching for the best value of the penalty term c. Instead, they search for an adversarial example in the Euclidean ball centered in x with radius ε by performing some gradient descent steps with the loss function used to train the model and project the result on the sphere. Depending on whether the solution is an adversarial example, they adjust the radius of the sphere and iterate the procedure. Similarly, the method *Fast Minimum Norm* (FMN) proposed in [10], a projection strategy suited also for different l_p norms. Another approach, namely *Augmented Lagrangian Method for Adversarial attacks* (ALMA) [11], uses the same paradigm but avoids searching for the best penalty c through a line-search, by exploiting the Lagrangian duality theory [28].

2.2.4 The deepfool method

Being the ideas presented in this chapter, strongly related to the findings and the motivations behind the DeepFool method, we briefly review this strategy. DeepFool (DF), [6], is a fast method for finding a minimal adversarial perturbation. It leverages the explicit formulation of the Euclidean distance with respect to the classification boundary in the linear case, to quickly generate accurate solutions for MAP. In short, the method provides an approximate solution of MAP by performing an iterative gradient-based algorithm with variable step size at each iteration. To be compliant with the terminology used in Section 2.3, the problem solved by DF can be rewritten by considering the minimal solution of a list of less expensive minimum problems $d(x,l) = \min_{j \neq l} d_j(x)$, where $d_j(x,l)$:

$$\begin{aligned} d_j(x,l) = \min_{\delta} \quad & \|\delta\| \\ \text{s.t} \quad & f_l(x+\delta) \leq f_j(x+\delta). \end{aligned} \tag{2.18}$$

The main idea consists of building a sequence $x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots$ that converges to an approximate solution of MAP, which lies in the adversarial region $\cup_{j \neq l} R_j$.

Given $x^{(k)}$, let $\tilde{f}_j(x)$ be the first order approximation of $(f_l(x) - f_j(x))$ in $x^{(k)}$. Then, the next element of the sequence $x^{(k+1)}$ is obtained by considering the minimal solution $d_j(x^{(k)}, l)$ of Problem 2.18 applied to $\tilde{f}_j(x)$ rather than $(f_l - f_j)(x)$. Since $\tilde{f}_j$ is an affine function, the problem has an exact solution of the form

$$x^{(k+1)} = x^{(k)} - \frac{\tilde{f}_j(x^{(k)})}{\|\nabla \tilde{f}_j(x^{(k)})\|} \frac{\nabla \tilde{f}_j(x^{(k)})}{\|\nabla \tilde{f}_j(x^{(k)})\|}. \tag{2.19}$$

Empirically speaking, the procedure reaches convergence in $K \approx 3$ steps, resulting in $2CK$ forward and backward passes, if applied to a classifier with C classes. However, it is crucial to point out that, since the iteration is stopped when the adversarial region is reached, there is no guarantee that the procedure will provide a solution of MAP. Indeed, the procedure just ensures that a feasible perturbation satisfying the constraint $\hat{k}(x+\delta) \neq l$, is found. In other words, to the best of our knowledge, there are no theoretical point-wise estimations of the approximation error, but only estimations of the average distance from the classification boundary [29].

2.2.5 The verification of DNNs

Verification methods aim at establishing whether, given a sample x and a bound $\varepsilon \geq 0$, each sample in the l_p-ball centered in x with radius ε is classified with the same class of x. Verification can be performed by solving the following problem

$$\begin{aligned} \min \quad & c^T z \\ \text{s.t.} \quad & \|\delta\|_p \leq \varepsilon \\ & z = f(x+\delta) \end{aligned} \quad , \tag{VP}$$

where c depends on the statement we want to verify. For example, given a class s other than the correct class l, we can determine the robustness of the sample x targeted to the class s only by looking at the sign of the solution where $c = e_l - e_s$. Precisely, if the solution of VP is positive, then the component z_s of the output score vector is lower than z_l in the neighborhood of radius ε.

Nevertheless, as proved in [12], Problem VP is NP-complete for ReLU networks, and hence formal *complete verification* strategies are unfeasible for commonly large networks. Other works [13], [30]–[32] bound the inner network activations to relax the constraints and provide an *incomplete verification*. However, being computationally expensive, these strategies do not scale to large networks and can be applied to multi-layer-perceptrons only or relatively small convolutional neural networks.

A different approach is given by [33], in which the authors search for the largest hyper-rectangle $\mathcal{S}_\varepsilon$, centered in x and with semi-sides of length $\varepsilon \in \mathbb{R}^n_+$, such that $\hat{k}(x+\delta) = l$ for each $\delta \in \mathcal{S}_\varepsilon(x)$. However, as in [13], solving such a problem requires estimating the bounds of the internal activations so that the method does not scale well to large networks. More scalable approaches have been provided in [34], [35]. [34] leverage dual optimization theory, which enables the verification of neural networks capable of accurately classifying images from the MNIST and CIFAR10 datasets. Recently, Wang et al. [35] proposed β-CROWN, which improves the computation of the inner activation bounds in the case of ReLU activation by splitting the verification problem into two easier problems based on the neuron outcome sign. Observe that the aforementioned strategies are suited for l_∞-bounded attacks and limit their analysis to ReLU activations only.

A scalable verification method based on Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER), is proposed by [36]. CLEVER considers l_2-bounded attacks and provides a lower bound β_L of $d(x,l)$ (as defined in Problem (MAP)) by evaluating the gradient of the network f on random samples in a neighborhood of x. However, the thightness of the bound strongly depends on the number of gradient evaluations: the higher the number of evaluations, the narrower the bound. Hence, achieving accurate results requires a long computational time.

2.3 MAP via root-finding algorithms

This section proposes two strategies that provide an approximate solution to the MAP problem by reducing it to the minimal root problem B-RMP presented in the Section 2.1. Since the formulation of the problem B-RMP concerns only the binary-classification case, in this section, we show how to reduce the multi-class case to the solution of one or multiple B-RMP. In detail, for each (x,l), a definition of $t(x,l)$ as an approximation of $d(x,l)$ will be given such that the following inequalities hold

$$\frac{1}{\rho} t(x,l) < d(x,l) < t(x,l). \tag{2.20}$$

Nevertheless, observe that the the extension of the binary case to a multi-class classifier is not unique, thus two methods will be proposed.

2.3.1 The closest boundary strategy

The *Closest Boundary* strategy (CB) leverages the idea that the minimum problem related to a classifier with C classes can be reduced to a list of minimum problems for binary classifiers. In detail, let x be a sample, correctly classified by f with label $l \in \{1,\dots,C\}$, and let

$$\begin{aligned} d_j(x,l) = \min_{\delta} \quad & \|\delta\| \\ \text{s.t} \quad & f_l(x+\delta) \le f_j(x+\delta). \end{aligned} \tag{2.21}$$

Then, we observe that $d(x,l) = \min_{j\neq l} d_j(x,l)$, where $d(x,l)$ solves MAP. This can be proved by reformulating the statement with the following inequalities

$$\min_{j\neq l} d_j(x,l) \leq d(x,l) \leq \min_{j\neq l} d_j(x,l).$$

Let $\delta^{(j)}$ be the solution of $d_j(x,l)$. The second inequality is a consequence from the fact that $\delta^{(j)}$ satisfies the constraint of MAP and that, by construction, $d(x,l)$ is lower than $\|\delta\|$ for each feasible δ. The first inequality, instead, can be proved by observing that Problem MAP is equivalent to

$$\begin{aligned} d(x,l) = \min_{\delta} \quad & \|\delta\| \\ \text{s.t} \quad & f_l(x+\delta) \leq \max_{j\neq l} f_j(x+\delta). \end{aligned} \tag{2.22}$$

Hence, if δ^* is the solution of Problem MAP and if $j^* \in \operatorname{argmax}_{j\neq l} f_j(x+\delta^*)$, then, by construction, δ^* satisfies the constraint of Problem 2.21 for j^*, and so $\min_{j\neq l} d_j(x,l) \leq d_{j^*}(x,l) \leq d(x,l)$. In conclusion, if $t_j(x,l)$ is the solution of B-RMP with $f(x) = f_l(x) - f_j(x)$, then $d(x,l)$ can be approximated by $t(x,l) = \min_{j\neq l} t_j(x,l)$.

More informally, if $B_{jl} := \{p \in \mathbb{R}^n : f_l(x) = f_j(x)\}$ is the classification boundary of the binary classifier $f_l - f_j$, we can reduce MAP to the problem of finding the closest intersection between the boundary B_{jl} and the straight line passing through x with the direction provided by the gradient of f. A good aspect of this strategy is that it reduces to the solution of a sequence of minimum problems by preserving the regularity of f, which indeed has a big impact on the accuracy of the approximation as shown in Section 2.1. For the sake of clarity, the procedure described above is summarized in Algorithm 1, where function `Zero`, called at Line 7, is any root finding algorithm for univariate functions that solves B-RMP.

Algorithm 1: Pseudocode implementing the Closest Boundary strategy depending on a root-finding algorithm.

```
   Data: Zero                                   !The root-finding algorithm.
   Input: $x$, $l$, $f$                         !The safe sample and the DNN.
   Output: $t$, $\nu$                           !The distance and the direction
1  t = $\infty$;
2  for $j = 1, \dots, c$ and $j \neq l$ do
3      $F(x) := f_l(x) - f_j(x)$;
4      grad = $\nabla F(x)$;
5      $\nu_j$ = -grad/$\|$grad$\|$;
6      $g(t) := F(x + t \cdot \nu_j)$;
7      $t_j$= Zero(g);
8      if $t_j < t$ then
9          $t = t_j$ ;
10         $\nu = \nu_j$;
11 return $t, \nu$;
```

2.3.2 Fast outer boundary strategy

The CB algorithm presented in the previous section can bring a large computational cost for a classifier f that distinguishes a large number of classes. In fact, if O_j is the amount of forward and backward passes required to compute each $t_j(x, l)$, then the total cost O can be estimated as $\sum_{j \neq l} O_j$. The *Fast Outer Boundary* strategy (FOB) is hence proposed here to contain the computational cost. The minimum problem MAP can be reduced to the minimal root problem B-RMP by considering $L(x, l) = f_l(x) - \max_{j \neq l} f_j(x)$ and observing that L acts like a binary classifier that takes positive values in the region R_l and negative values in the outer region $\cup_{j \neq l} R_j$. Hence, the approximation of $d(x, l)$ can be deduced by solving the minimal root problem obtained by substituting f with L in Problem B-RMP. Observe that, differently from the previous strategy, this one requires the solution of a single minimal root problem. The pseudocode formulation of the FOB strategy can easily be obtained as a variant of Algorithm 1 by replacing F with L and removing the `for` loop.

2.3.3 Root-finding algorithms

Both strategies leverage two main observations: (i) the gradient of f suggests the fastest direction to reach the adversarial region; and (ii) due to the objective function, the minimal perturbation lays on the classification boundary. The two considerations above naturally bring to searching the minimal perturbation as the intersection between the classification boundary and the direction of the gradient ∇f, that directly brings to root-finding problem. In this work, the FOB and CB methods are tested by solving the root problem B-RMP with a customized version of the *Bisection Algorithm* and the vanilla *Newton Algorithm*, which return the approximate distance $t(x, l)$ for each sample (x, l). The bisection method has been adapted to better fit the task. A more detailed illustration is provided below. In general, the bisection method allows finding a zero of a scalar univariate continuous function $g : [a, b] \to \mathbb{R}$ under the assumption that $g(a) > 0$ and $g(b) < 0$, without requiring the computation of the derivative of g. Note that in our case $a = 0$ because in Problem B-RMP the variable t is positive.

Solving B-RMP requires finding the minimal positive root of the g function, which, in general, is not a solution of the vanilla bisection algorithm. In fact, in the searching interval $[0, b]$, function g is not guaranteed to be monotone and it can change sign, from positive to negative and vice-versa. To tackle this issue, we apply a pre-processing to the initial searching interval $[0, b]$ that is inspired by Armijo rule for line search methods [22]. In detail, given a maximum number of attempts R, we consider $\tilde{b} = b \cdot 2^{-\tilde{k}}$, where

$$\tilde{k} = \max \left\{ i \in \mathbb{N} \, : \, g(b \cdot 2^{-i}) < 0, \, i = 0, 1, \cdots, R \right\} \tag{2.23}$$

and we start the bisection in $[0, \tilde{b}]$. The pseudocode that implements the Closest Boundary strategy is shown in Algorithm 2. Line 20 reduces the number of forward passes of the model by stopping the inner iteration if the lower bound `t_curr_low` of the current label is higher than the actual overall minimal estimation `t`.

Algorithm 2: Pseudocode for bisection algorithm, with Armijo-like upper bound estimation, applied to the Closest Boundary strategy.

```
Data: t_up, MaxIter, MaxAttempt
Input: $x$, $l$, $f$                                   !The sample and the DNN
Output: $t$,$\nu$                                      !The distance and the direction
Tol = $5 \cdot 10^{-5}$;
t = $\infty$;
for $j = 1, \dots, c$ and $j \neq l$ do
    $F(x) := f_l(x) - f_j(x)$;
    grad = $\nabla F(x)$;
    $\nu_j$ = $-$grad/$\|$grad$\|$;
    !Starting of the bisection algorithm;
    t_curr_low = 0;
    t_curr_up = Armijo(g, b=t_up);
    for step = 1,..., MaxIter do
        t_curr = (t_curr_low + t_curr_up )/2;
        x_curr = x + t_curr * $\nu_j$;
        out = F(x_curr);
        if out > 0 then
            t_low = t_curr;
            out_low = o;
        else
            t_up = t_curr;
            out_up = o;
        if t_curr_low > t then
            Break                      !Reduce the amount of iterations;
        if 0 > o_up > -Tol then
            Convergence;
    if t_up < t then
        $t = t\_up$ ;
        $\nu = \nu_j$;
return $t$, $\nu$;
```

Table 2.1: Summary of the most frequent symbols.

Symb.	Dimensionality	Meaning
f	$: \mathbb{R}^n \to \mathbb{R}^C(\mathbb{R})$	classifier (binary classifier)
$d(x,l)$	$\in \mathbb{R}$	solution of MAP
$t(x,l)$	$\in \mathbb{R}$	solution of B-RMP
$\mathcal{B}$	$\subseteq \mathbb{R}^n$	classification boundary (binary classif.)
Ω_σ	$\subseteq \mathbb{R}^n$	tubular neighborhood of $\mathcal{B}$ of radius σ
ρ	$\in \mathbb{R}$	coefficient of Inequality 2.13
$\sigma(\rho)$	$\in \mathbb{R}$	radius in which Inequality 2.13 holds.
$\mathcal{K}_f(\Omega)$	$\in \mathbb{R}$	boundary proximity index
σ^*	$\in \mathbb{R}$	significant σ
$\hat{\sigma}^*$	$\in \mathbb{R}$	empirical estimation of σ^*

2.3.4 Error estimation for multi-class classifiers

We conclude this section observing that, the theoretical analysis presented in Section 2.1 can be extended to a multi-class classifier by leveraging the two methodologies presented in Section 2.3. If $f : \mathbb{R}^n \to \mathbb{R}^C$ is a classifier with C classes, both strategies reduce to a search for a solution of the minimal root problem B-RMP for one or more binary classifiers in which the analysis can be applied.

The Fast-Outer-Boundary strategy presented in Section 2.3.2 consists in solving Problem B-RMP for a binary classifier of the form $L^{(l)} : \mathbb{R}^n \to \mathbb{R}$ where $L^{(l)}(x) := L(x,l) = f_l(x) - \max_{j \neq l} f_j(x)$. Thus, by applying Theorem 1 to $L^{(l)}$, we deduce the existence of a $\sigma^{(l)}(\rho)$ such that the estimation holds for each sample x with $\hat{k}(x) = l$. Therefore, by considering $\sigma(\rho) = \min_l \sigma^{(l)}(\rho)$, we obtain a radius for which the Equation (2.20) holds.

The Closest-Boundary strategy presented in Section 2.3.1 consists instead in solving Problem B-RMP for a list of minimal root problems relative to binary classifiers of the form $f_{jl} = f_l - f_j$. In particular, for each ρ, Theorem 1 ensures the existence of a neighborhood with radius $\sigma_{jl}(\rho)$ such that the following inequalities holds

$$\frac{1}{\rho} t_j(x,l) \leq d_j(x,l) \leq t_j(x,l), \forall j,$$

where we keep the notation of Section 2.3.1. By taking the minimum over $j \neq l$ we deduce the estimation in Equation (2.20) for every x with $\hat{k}(x) = l$ and $x \in \Omega_{\sigma_l(\rho)}$, where $\sigma_l(x) = \min_{j \neq l} \sigma_{jl}(\rho)$. In conclusion, by considering $\sigma(\rho) = \min_{l,j \neq l} \sigma_{jl}(\rho)$, we deduce an extension of the desired inequality for the multi-class case.

2.4 Empirical results

This section presents a set of experiments aimed at validating the strategies proposed in Section 2.3. They are executed on four neural classifiers, each trained on a different dataset. The approximate distances provided by the tested strategies are compared in Section 2.4.3 with the *Iterative Penalty* method (Section 2.4.1), in order to provide a ground-truth distance. Section 2.4.4 reports an empirical estimation of σ for three noticeable values of ρ. Finally, Section 2.4.5 discusses the case in which all the classifiers are attacked with different known methods. The magnitude of each attack is bounded to be lower than $t(x)/\rho^*$ in order to show that the attack success rate drops to about zero for samples in $\Omega_{\hat{\sigma}^*}$, where $\hat{\sigma}^*$ is an estimation of σ^*.

2.4.1 Groundtruth distance estimation

In order to compare the approximate distances that solve Equation (B-RMP), we need an accurate measure of the theoretical distance $d(x)$, which is practically unknown in the general case. To tackle this problem, based on the ideas presented in [37] and [7], we solve Equation (MAP) by reducing to the following minimum problem with penalty analogous to Equation (2.17)

$$d(x, l; c) = \min_{\delta \in \mathbb{R}^n} \quad \|\delta\| + c \cdot L(x + \delta, l)^+, \tag{2.24}$$

where $L(x, l) = f_l(x) - \max_{j \neq l} f_j(x)$ and $L^+ = \max\{0, L\}$. For each sample (x, l) and for each penalty value c, we perform a gradient descent with the Adam optimizer [38], using the default parameters, up to 10^4 iterations, stopping the procedure when $-\texttt{Tol} < L(x^{(k)}, l) \leq 0$, where the tolerance $\texttt{Tol}$ is set to $5 \cdot 10^{-5}$. Note that this convergence criterion ensures that the solution lies close to the boundary and it is contained in the adversarial region $\cup_{j \neq l} R_j$. Similarly to [7], the best penalty c is selected through a bisection-like search. In details, let $c_{\texttt{low}} = 0$ and $c_{\texttt{up}}$ such that $d(x, l; c_{\texttt{low}}) = 0$ and $d(x, l; c_{\texttt{up}})$ does not converge for all the samples x in the dataset. In our experiments, we discovered that $c_{\texttt{up}} = 100$ is large enough to satisfy this definition. Then, through successive bisections, we can define $c_{\texttt{curr}} = \frac{1}{2}(c_{\texttt{low}} + c_{\texttt{up}})$ and either (i) set $c_{\texttt{up}} = c_{\texttt{curr}}$ (i.e., decreasing $c_{\texttt{up}}$) if the optimization for $d(x, l; c_{\texttt{curr}})$ does not converge, or (ii) set $c_{low} = c_{\texttt{curr}}$ (i.e., increasing c_{low}) if it converges. We stop the search for c after 12 bisections. The whole procedure is implemented in batch mode to exploit GPU acceleration. During the experiments, we noted that the Iterative Penalty (IP) method can provide, for a few of the tested samples, an estimation of $d(x)$ that is slightly higher than other global methods, such as DeepFool (DF)[6] and Decoupling Direction Norm (DDN) [9]. Thus, in order to adopt a more precise ground truth, we decided to consider for each sample x the ground-truth distance $d(x)$ as the minimum distance obtained with IP, DF, and DDN.

2.4.2 Experimental Settings

The proposed techniques were evaluated on four different datasets, each associated with a different neural network. In the following, we use the name of the dataset to refer to the experimental setting composed of the dataset itself and the corresponding network.

MNIST

The MNIST handwritten digits dataset, introduced in [39], was used to train a vanilla LeNet within a 2×2-MaxPool, 2 convolutional, and 3 fully connected layers, achieving a 1% error rate on the test set. The training was performed without data augmentation, using the Adam optimizer [38] (default hyper-parameters) to minimize the Cross-Entropy Loss with a 128 batch size for 5 epochs.

Fashion MNIST

This dataset includes 50,000 training images and 10,000 test images (28×28 greyscale pixels) grouped in 10 classes [40]. Compared to MNIST, this dataset is less trivial and requires finer tuning to craft a model with good accuracy. It was used to train a vanilla LeNet with the same structure as the previous one. The training was performed without data augmentation, by minimizing the Cross Entropy loss with the Adam optimizer for 30 epochs (with a batch size of 128) to achieve a 91% accuracy on the test set.

CIFAR10

This dataset contains 60,000 RGB images of size 32×32 pixels divided into 10 classes [41]. Inspired by [42], it was used to train a *Resnet32* model [43] over the first 50,000 images of the dataset with data augmentation, as described in the original paper. In detail, the images were randomly cropped and horizontally flipped. The training was performed by minimizing the Cross-Entropy loss for 182 epochs by the *stochastic gradient descent with Nesterov momentum* (SGD) [44] with a starting learning rate of 0.1, momentum of 0.9, and a weight decay of $1e-4$. The learning rate was decreased using a multiplicative factor of 0.1 after the 90th and the 135th epoch, achieving a 8.8% error rate over the test set. This is in line with the original results of [42].

GTSRB

The *German Traffic Sign Recognition Benchmark* [45] contains about 51,000 traffic signs RGB images of various shapes (from 15×15 to 250×250), grouped in 43 classes. It was used to train a *MicronNet* [46], a compact network similar to LeNet that classifies pixel-wise standardized 48×48 images. The training was performed over the first chunk of the dataset, containing $\approx 39{,}000$ images with a data augmentation technique. During training, each image was randomly rotated by an angle in $\pm 5°$, translated towards a random direction with magnitude lower than 10%, and finally scaled with a factor between 0.9 and 1.1. Each transformed image was then scaled to have a dimension of 48 pixels per side. The model was trained to minimize the Cross Entropy loss by the SGD optimizer with a learning rate of $7e-3$, a momentum of 0.8, and a weight decay of $1e-5$, for 100 epochs. The learning rate was decreased every 10 epochs with a multiplicative factor of 0.9. We achieved a 1.2% error rate over the test set, which is comparable with the state-of-the-art classification performance with this dataset.

2.4.3 Comparing distances

This section focuses on comparing the estimated distances to the ground-truth distance for the four network models and corresponding data sets. For each sample (x, l), the approximate distances $t(x, l)$ are obtained by applying the zero finding algorithms (Bisection and Newton) to the strategies CB and FOB presented in Section 2.3. The ground-truth distance $d(x, l)$ is computed through the technique presented in Section 2.4.1. Note that, in this section we only focus on the unconstrained formulation of the MAP (hence without box-constraint), since the aim is to support the theoretical funding of Section 2.1.

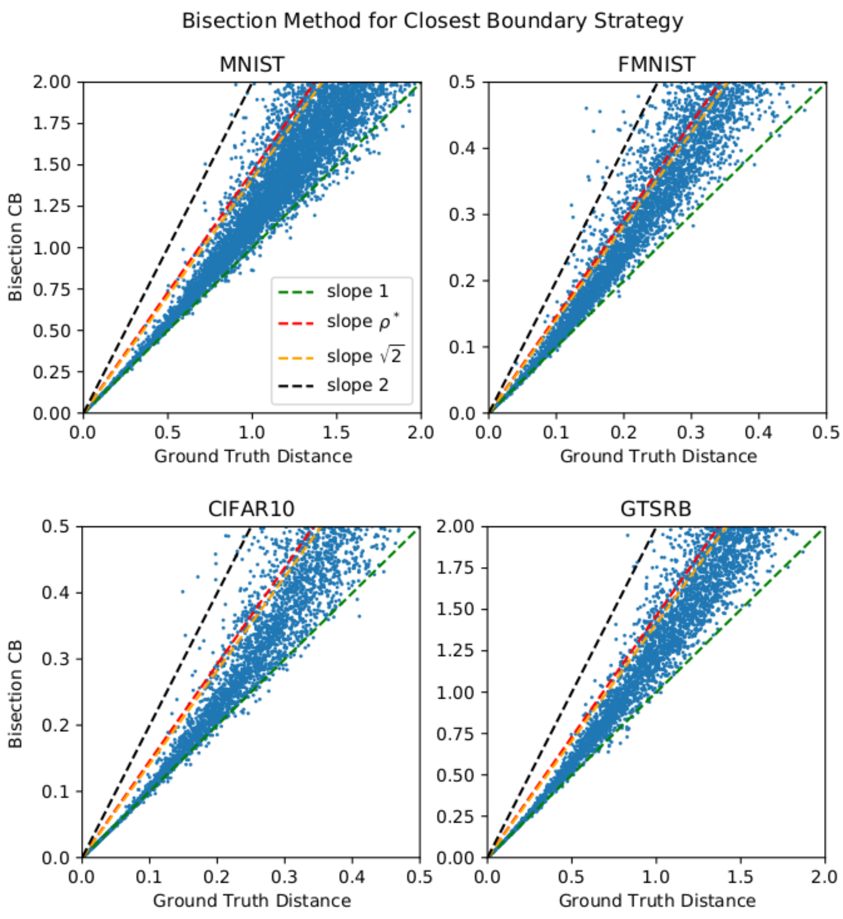


Figure 2.3: Comparison of the approximate distance $t(x, l)$, computed by the Bisection CB strategy, and the ground-truth distance $d(x, l)$, for the four models considered in Section 2.4.2. Each dot represents the pair $(d(x, l), t(x, l))$ where x is a sample with label l. The region between the green line (slope 1) and the other lines (slope $\sqrt{2}$, ρ^*, 2), highlights the samples for which the Inequality 2.20 holds. Observe that, according to the theoretical results, the closer the boundary (small $d(x, l)$) the higher the number of dots in the region of interest.

Figure 2.3 shows a comparison between the approximate distance $t(x, l)$, computed by the Bisection CB strategy, and the ground-truth distance $d(x, l)$ for the four models considered in Section 2.4.2. For each sample x of label l, each dot in a graph represents the pair $(d(x, l), t(x, l))$. The dashed green line with slope 1 represents the points in which $d(x, l) = t(x, l)$. The other three lines have slopes $\sqrt{2}$, ρ^* and 2, where ρ^* is defined in Observation 3, and represent Equation (2.20) for different values of ρ. Observe that almost all the points close to the boundary (i.e., those with a small ground-truth distance to the boundary) are located above the green line and below the others, confirming that the estimation $t(x) \leq \rho d(x)$

		MNIST		FMNIST		CIFAR10		GTSRB	
		Avg.Dist.	# Evals	Avg.Dist.	# Evals	Avg.Dist.	# Evals	Avg.Dist.	# Evals
Strategy	Algorithm								
FOB	Bisection	1.645	17	0.455	16	0.508	17	2.221	17
CB	Bisection	1.467	17*†	0.385	16*†	0.483	17*†	1.667	17*†
FOB	Newton	1.641	3	0.442	3	0.496	4	2.169	4
CB	Newton	1.466	3*	0.385	3*	0.481	3*	1.668	3*
DF		1.526	2*	0.318	3*	0.346	3*	1.516	3*
DDN		1.287	1000	0.281	1000	0.338	1000	1.343	1000
IP		1.198	30162	0.261	32774	0.289	50609	1.262	34769
GT		1.172	-	0.255	-	0.283	-	1.204	-

* Average number of evaluations for each class of the datasets.
† Only one backward for each run. The remaining evaluations just perform forwards of the model.

Table 2.2: Average distance from the boundary and average number of evaluations of the models for the four datasets obtained with different methods. The behavior of the tested methods for samples close to the boundary is detailed in Figure 2.4. Columns '# Evals' report the number of times the method requires a forward and a backward pass through the model.

	MNIST	FMNIST	CIFAR10	GTSRB
n_1	722	1198	849	576
n_2	2200	1723	1311	1347
n_3	3669	1767	1628	1696
n_4	2184	1350	1636	1866
Tot.	8775	6038	5424	5485

Table 2.3: Summary of the number of samples in each interval for each dataset. The MNIST and GTSRB are partitioned into intervals of length 0.5 from 0 to 2. The FMNIST and CIFAR10 are partitioned into intervals of length 0.125 from 0 to 0.5.

holds.

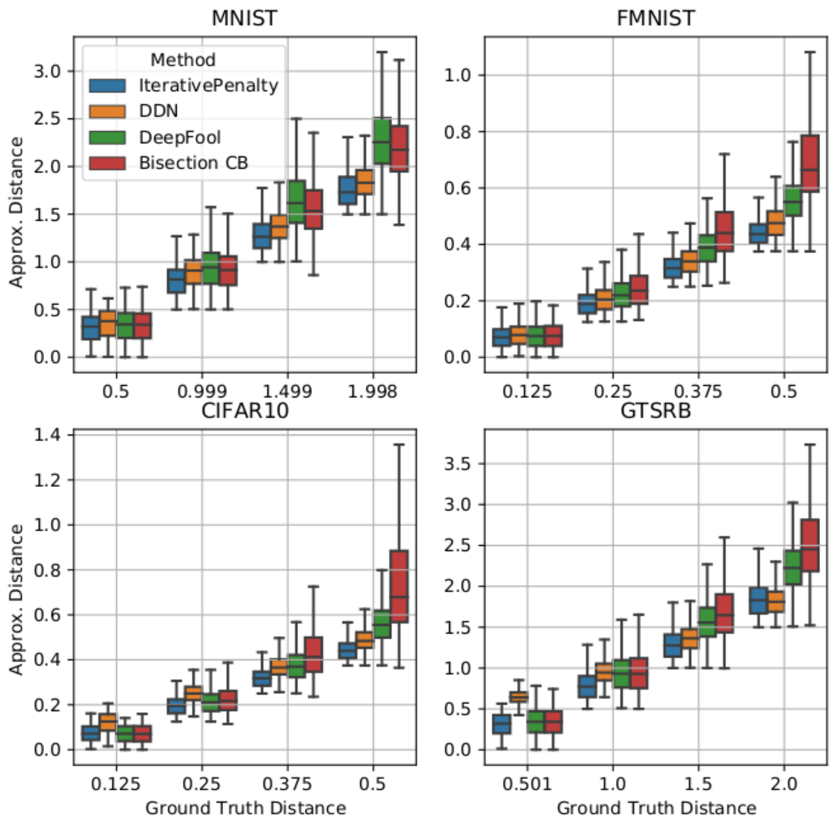


Figure 2.4: Comparison of the distances computed by the Bisection CB strategy, DeepFool, Decoupling Direction Norm (DDN), and Iterative Penalty with respect to the ground-truth distance. For a clearer representation, the ground-truth distance is partitioned into four intervals that contain a number of samples summarized in Table 2.3. For each method, the lower and the upper side of each box represent the first and the fourth quartile Q_1 and Q_2, respectively; the lower and the upper whisker represent the quantiles $Q_1 - 1.5 \cdot I_q$ and $Q_3 + 1.5 \cdot I_q$, respectively, where I_q is the interquartile range.

Table 2.2 reports the average distances from the boundary for each dataset and for each tested strategy and the average number of evaluations for a timing comparison. The statistics are computed over all the samples in the test set that satisfy the following conditions:

(i) the sample is correctly predicted by the model;

(ii) the algorithms reach the convergence;

(iii) the ground-truth distance is lower than 2.0 for MNIST and GTSRB, and lower than 0.5 for FMNIST and CIFAR10 (threshold values were selected to include a large part of the test set while still focusing on the region close to the boundary).

The amount of tested samples is detailed in Table 2.3. As one may expect, DF, DDN and Iterative Penalty (IP) provide lower distances with respect to our strate-

			MNIST	FMNIST	CIFAR10	GTSRB
ρ	Algo.	Strategy				
$\sqrt{2}$	B	FOB	0.34	0.06	0.04	0.15
		CB	0.37	0.06	0.13	0.58
	N	FOB	0.34	0.06	0.04	0.15
		CB	0.37	0.01	0.00	0.58
ρ^*	B	FOB	0.37	0.06	0.04	0.15
		CB	0.59	0.08	0.13	0.58
	N	FOB	0.37	0.06	0.04	0.15
		CB	0.59	0.01	0.00	0.58
2	B	FOB	0.37	0.12	0.11	0.49
		CB	0.72	0.12	0.17	0.87
	N	FOB	0.37	0.12	0.11	0.49
		CB	0.72	0.01	0.00	0.87

Table 2.4: Comparison of all the $\hat{\sigma}$ estimated by the different techniques.

gies CB and FOB. However, the distances computed by CB and FOB are associated with a bound on the approximation error relative to the theoretical distance $d(x)$. The boxplot in Figure 2.4 provides a comparison of the approximate distances computed by the Bisection method applied to the CB strategy, DeepFool, IP, and DDN. The ground truth distance reported on the x-axis is partitioned, differently for each dataset, into four intervals, whose dimensions are summarized in Table 2.3. Again, note that for points near the boundary, our method provides an accurate estimation of d, whereas, far from the boundary, global techniques result in being more accurate, returning a better approximation of the ground-truth distance.

2.4.4 Empirical estimation of the tubular neighborhood

Theoretically, Theorem 1 ensures that for each $\rho \in (\sqrt{2}, 2)$ there exists a $\sigma(\rho)$ for which Inequality (2.20) holds. In practice, however, for an arbitrary classifier f, such a $\sigma(\rho)$ cannot be deduced explicitly. Nevertheless, we can empirically estimate its value. In particular, given a data set $\mathcal{X}$, we can define $\hat{\sigma}(\rho)$, an estimation of $\sigma(\rho)$, as follows:

$$\hat{\sigma}(\rho) = \min\left\{ d(x,l) \, : \, \frac{t(x,l)}{d(x,l)} > \rho, \quad (x,l) \in \mathcal{X} \right\}, \tag{2.25}$$

which corresponds to the maximum distance for which Inequality (2.20) holds for the samples in $\mathcal{X}$.

Table 2.4 reports different estimations of σ for different values of ρ, in accordance with Section 2.4.3. For each ρ, the estimation $\hat{\sigma}(\rho)$ is deduced on a subset of the testset built by randomly sampling 60% of the images. Observe that the values of σ provided by CB are larger than or equal to those provided by FOB. In terms of algorithms, the customized bisection algorithm (augmented with the Armijo-like rule) provides more reliable results with respect to the Newton method. We believe this is due to the fact that *there is no guarantee that the Newton algorithm provides the smallest positive zero of the function.* As the theoretical value of σ (that depends on the boundary proximity index), also the empirical estimations can be seen as well as a measure of the regularity of the models: the higher $\hat{\sigma}$, the higher the regularity of the model (or the boundary). Also, observe that these results are in line with Table 2.2, in which the model for FMNIST has an average distance that is lower than the one of the LeNet for MNIST (on which the images have the same dimension and have been normalized with the same mean and standard deviation).

2.4.5 Adversarial robustness in the tubular neighborhood

This section evaluates the goodness of the empirical estimation $\hat{\sigma}^*$ of the theoretical σ^* (defined in Observation 3) to assess the model robustness against adversarial examples bounded in magnitude by $t(x)/\rho^*$. In formulas, let $\tilde{x} = Adv_\varepsilon(x,l)$ an adversarial example crafted with an unknown attack technique Adv_ε that for each sample (x,l) provides a new sample $\tilde{x}$ (if exists) such that $\hat{k}(x) \neq l$ and $\|\tilde{x}-x\| \leq \varepsilon$. We want to empirically show that

$$\left\{x \in \Omega_{\sigma^*} \,:\, \exists Adv_\varepsilon(x,\hat{k}(x);\varepsilon),\, \varepsilon < \frac{t(x)}{\rho^*}\right\} = \emptyset. \tag{2.26}$$

In other words, we empirically assess that for each sample distant from the boundary less than $\hat{\sigma}^*$, there are no adversarial perturbations with a magnitude smaller than $t(x,l)/\rho^*$. For this purpose, we only test the approximation $t(x)$ provided by the CB strategy with the bisection method. In fact, higher values of $\hat{\sigma}$ represent a worst case to be tested, since there are more samples with a distance lower than $\hat{\sigma}$. By using FoolBox [47], we generated adversarial examples for the four datasets with the following techniques: *Decoupling Norm Direction* (DDN) [9], *Deep Fool* (DF) [6], *Projected Gradient Descent* (PGD) [16], *Fast Gradient Method* (FGM) [47]. For each dataset $\mathcal{X}$, and for each sample $(x,l) \in \mathcal{X}$, we considered the `clipped` output of FoolBox that is guaranteed to have magnitude lower than ε, i.e. $\|\tilde{x}-x\| < \varepsilon$. Observe that in this test the magnitude of the attack ε is never computed by using the ground-truth distance $d(x,l)$, but by setting $\varepsilon = t(x,l)/\rho^*$.

The results of this experiment for the four datasets are shown in Figure 2.5, in which each graph reports the number of adversarial examples found with magnitude $t(x)/\rho^*$ as a function of the ground-truth distance $d(x,l)$. In detail, each stepped line reports, as a function of d, the cardinality of the set

$$\left\{(x,l) \in \mathcal{X} \,:\, \exists Adv_\varepsilon(x,l),\, \varepsilon = \frac{t(x)}{\rho^*},\, d(x,l) \leq d\right\}$$

rescaled to be one for the maximum value of d, i.e., the fraction of points that are out of the bound for the tested attack. All graphs show that the higher d, the higher the number of samples that escape the bounds (a sample escapes the bounds if $t(x,l)/\rho^*$ is higher than the real distance from the boundary). In each plot, the values of $\hat{\sigma}^*$ computed in Table 2.4 are represented by the dashed red lines. It is important to observe that the estimation of $\hat{\sigma}^*$ was deduced as explained in the previous section, i.e., by applying Equation (2.25) without knowing the results of the attacks in advance.

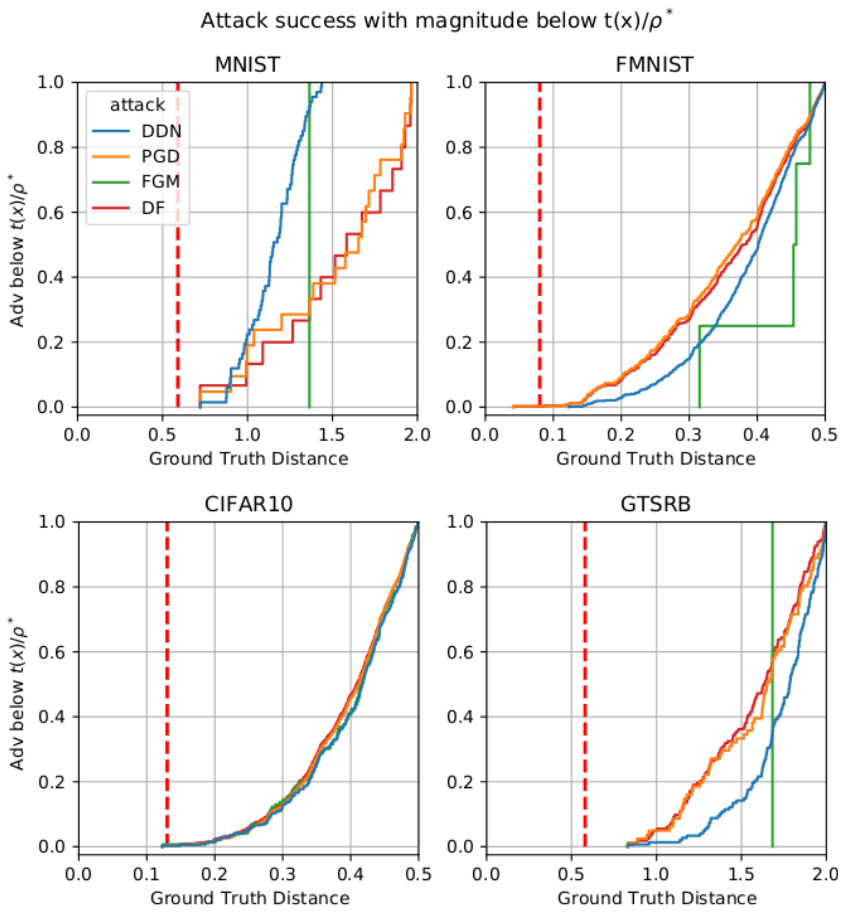


Figure 2.5: Attack success rate cumulative curve for attacks bounded in magnitude less than $t(x)/\rho^*$ obtained with Bisection method and Closest Boundary strategy. The dashed red line represent $\hat{\sigma}^*$, which approximates σ^* of Theorem 1. For MNIST and GTSRB, none of the samples with a distance from the boundary lower than $\hat{\sigma}^*$ can be perturbed by the tested bounded attacks, in accordance with the theoretical results. For the FMNIST and the CIFAR10 dataset, instead, the estimation $\hat{\sigma}^*$ results to be less accurate, failing in a tiny portion of the tested samples (3 and 5 samples overall respectively).

The result of this test shows that the two datasets FMNIST and CIFAR10 have a different behavior with respect to MNIST and GTSRB. In particular, for MNIST and GTSRB, the estimation of σ^* is more selective, meaning that the estimation done by Inequality (2.20) holds for distances slightly larger than $\hat{\sigma}^*$. Moreover, for FMNIST and CIFAR10 datasets, the estimation of σ^* results to be less accurate, and for a few samples (1 sample for each dataset) the attacks succeed even if the ground truth distance is lower than $\hat{\sigma}$, proving that the estimation in Inequality 2.20 does not hold in a neighborhood of radius $\hat{\sigma}^*$ at least for one example.

2.4.6 Comparison with CLEVER

This section compares the estimation $t(x, l)/\rho^*$, obtained by the bisection method with the CB strategy, with the lower bound β_L obtained by CLEVER implemented by IBM in [48]. Note that both CLEVER and the CB strategy can provide estimates of different quality depending on the number of evaluations of the model. To fairly compare the quality of the results provided by the two methods, it is hence required to bound the maximum number of evaluations that they perform. In our experiments, this bound was set to 20, which was empirically selected by observing that the CB strategy requires only one gradient evaluation and (on average) at most 17 forward passes per class to converge. Note that the recommended amount of gradient evaluations of CLEVER is $500 * 1024$, which is clearly far from the bound we imposed. It is also worth remembering that, for a given sample, the Bisection method with CB strategy only performs *one* gradient evaluation at the first step (for each class), while all the following steps of the algorithm only require a forward pass of the model (and no gradient computations).

	Avg		Failures [%]		Evals [#]		$\#\{d \leq \hat{\sigma}^*\}$
	t/ρ^*	β_L	t/ρ^*	β_L	t/ρ^*	β_L	
Dataset							
MNIST	0.27	0.35	0.21	32.27	14.73	20.00	970
FMNIST	0.03	0.04	0.14	42.66	16.10	20.00	715
CIFAR10	0.05	0.07	0.12	48.94	17.04	20.00	852
GTSRB	0.28	0.39	0.54	66.58	15.40	20.00	745

Table 2.5: Comparison between the lower bound t/ρ^* and β_L of CLEVER. Only samples with ground truth distance lower than $\hat{\sigma}^*$ for each dataset are considered. Columns "Failures[%]" summarize the percentage of samples for which the estimated lower bounds are not lower than the ground-truth.

Table 2.5 provides a close comparison between the two lower bounds $t(x,l)/\rho^*$ and β_L. The metric named "Failure [%]" represents the percentage of samples for which the expected lower bounds are higher than $d(x,l)$. The metric named "Eval [#]" counts the mean number of evaluations of the models for each class. Observe that, for all the datasets, the amount of failures of the CB strategy is much lower than the one of CLEVER (β_L). To consider only the scenarios supported by our theoretical analysis from Section 2.1, only samples with a ground-truth distance lower than the corresponding empirical lower bound $\hat{\sigma}^*$ are considered.

Chapter 3

1-Lipschitz Deep Neural Networks

As introduced in the Chapter 2, modern artificial neural networks achieve high accuracy and sometimes superhuman performance in many different tasks, but it is widely recognized that they are not robust to tiny and imperceptible input perturbations [1], [2] that, if properly crafted, can cause a model to produce the wrong output. Such inputs, known as *Adversarial Examples*, represent a serious concern for the deployment of machine learning models in safety-critical systems [49]. For this reason, the scientific community is pushing towards *guarantees* of

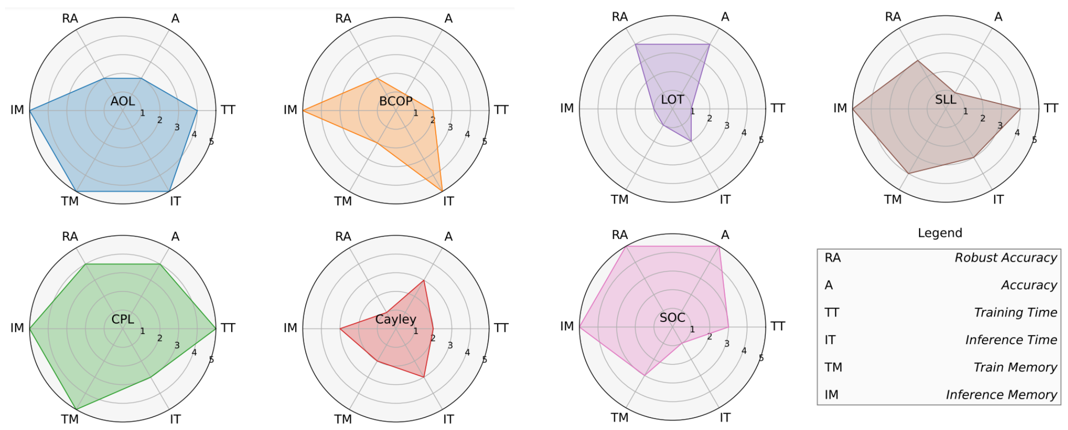


Figure 3.1: Evaluation of 1-Lipschitz methods on different metrics. Scores are assigned from 1 (worst) to 5 (best) to every method based on the results reported in Sections 3.2 and 3.4.

robustness. Recently, in the context of image classification, various approaches have been proposed to achieve certifiable robustness, including *Verification, Randomized Smoothing*, and *Lipschitz bounded Neural Networks*.

Verification strategies, that have been deepened in the Chapter 2, aim to establish, for any given model, whether all samples contained in a l_2-ball with radius ε and centered in the tested input x are classified with the same class as x. In the exact formulation, verification strategies involve the solution of an NP-hard problem [12]. Nevertheless, even in a relaxed formulation, [13], these strategies require a huge computational effort [36].

Randomized Smoothing (RS), initially presented in [50], represent an effective way of crafting a certifiable-robust classifier g based on a base classifier f. If combined with an additional denoising step, they can achieve state-of-the-art levels of robustness, [51]. However, since they require multiple evaluations of the base model (up to 100k evaluations) for the classification of a single input, they cannot be used for real-time applications.

Finally, Lipschitz Bounded Neural Networks represent a valid alternative to produce certifiable classifiers, since they only require a single forward pass of the model at inference time [18], [52]–[57]. Indeed, the difference between the two largest output components of the model directly provides a lower-bound, in Euclidean norm, of the minimal adversarial perturbation capable of fooling the model. Lipschitz-bounded neural networks can be obtained by the composition of 1-Lipschitz layers [58]. The process of parameterizing 1-Lipschitz layers is fairly straightforward for fully connected layers. However, for convolutions — with overlapping kernels — deducing an effective parameterization is a hard problem. Indeed, the Lipschitz condition can be essentially thought of as a condition on the Jacobian of the layer. However, the Jacobian matrix can not be efficiently computed.

In order to avoid the explicit computation of the Jacobian, various methods have been proposed, including parameterizations that cause the Jacobian to be (very close to) orthogonal [55], [56], [59], [60] and methods that rely on an upper bound on the Jacobian instead [54]. Those different methods differ drastically in training and validation requirements (in particular time and memory) as well as empirical performance. Furthermore, increasing training time or model sizes very often also increases the empirical performance. This makes it hard to judge from the existing literature which methods are the most promising. This becomes even worse when working with specific computation requirements, such as restrictions on the available memory. In this case, it is important to choose the method that better suits the characteristics of the system in terms of evaluation time, memory usage as well and certifiable-robust-accuracy.

This chapter aims at giving a comprehensive comparison of different strategies for crafting 1-Lipschitz layers from both a theoretical and practical perspective. For the sake of fairness, we consider several metrics such as *Time* and *Memory* requirements for both training and inference, *Accuracy*, as well as *Certified Robust Accuracy*. The main contributions are the following:

- An empirical comparison of 1-Lipschitz layers based on six different metrics, and three different datasets on four architecture sizes with three time constraints.
- A theoretical comparison of the runtime complexity and the memory usage of existing methods.

3.1 1-Lipschitz Layers

In recent years, various methods have been proposed for creating artificial neural networks with a bounded Lipschitz constant. The *lipschitz constant* of a function

$f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to the l_2 norm is the smallest L such that for all $x, y \in \mathbb{R}^n$

$$\|f(x) - f(y)\|_2 \leq L\|x - y\|_2. \tag{3.1}$$

This definition can be extended to networks and layers, by considering the l_2 norms of the flattened input and output tensors in Equation (3.1). A layer is called 1-Lipschitz if its Lipschitz constant is at most 1. For linear layers, the Lipschitz constant is equal to the *spectral norm* of the weight matrix that is given as

$$\|M\| = \sup_{\mathbf{v} \neq 0} \frac{\|M\mathbf{v}\|_2}{\|\mathbf{v}\|_2}. \tag{3.2}$$

A particular class of linear 1-Lipschitz layers are ones with an orthogonal Jacobian matrix. The Jacobian matrix of a layer is the matrix of partial derivatives of the flattened outputs with respect to the flattened inputs. A matrix M is orthogonal if $MM^\top = I$, where I is the identity matrix. For layers with an orthogonal Jacobian, Equation (3.1) always holds with equality and, because of this, a lot of methods aim at constructing such 1-Lipschitz layers.

The analysis of this chapter has been conducted by considering neural networks composed of 1-Lipschitz parameterized layers and 1-Lipschitz activation functions, with no skip connections and no batch normalization. Even though the commonly used ReLU activation function is 1-Lipschitz, Anil *et al.* [58] showed that it reduces the expressive capability of the model. Hence, we adopt the MaxMin activation proposed by the authors and commonly used in 1-Lipschitz models. Concatenations of 1-Lipschitz functions are 1-Lipschitz, so the networks analyzed are 1-Lipschitz by construction.

3.1.1 Spectral norm and orthogonal projection

Many recently proposed methods rely on a way of parameterizing orthogonal matrices or parameterizing matrices with bounded spectral norm. In this section, we discuss the fundamental methods for estimating the spectral norms of linear and convolutional layers, i.e. *Power Method* [61] and *Fantistic4* [62], and for crafting orthogonal matrices, i.e. Bjorck & Bowie [63], in Section 3.1.1.

Bjorck & Bowie [63] introduced an iterative algorithm that finds the closest orthogonal matrix to the given input matrix. In the commonly used form, this is achieved by computing a sequence of matrices using

$$A_{k+1} = A_k \left(I + \frac{1}{2} Q_k \right), \quad \text{for } Q_k = I - A_k^\top A_k \tag{3.3}$$

where $A_0 = A$, is the input matrix. The algorithm is usually truncated after a fixed number of steps, during training often 3 iterations are enough, and for inference more (e.g. 15) iterations are used to ensure a good approximation. Since the algorithm is differentiable, it can be applied to construct 1-Lipschitz networks as proposed initially in [58] or also as an auxiliary method for more complex strategies [55].

Power Method The *power method* was used in [61], [57] and [64] in order to bound the spectral norm of matrices.

It starts with a random initialized vector $\mathbf{u}_0$, and iteratively applies the following:

$$\mathbf{v}_{k+1} = \frac{W^\top \mathbf{u}_k}{\|W^\top \mathbf{u}_k\|_2}, \quad \mathbf{u}_{k+1} = \frac{W\mathbf{v}_{k+1}}{\|W\mathbf{v}_{k+1}\|_2}. \tag{3.4}$$

Then the sequence σ_k converges to the spectral norm of W, for σ_k given by

$$\sigma_k = \mathbf{u}_k^\top W \mathbf{v}_k. \tag{3.5}$$

This procedure allows us to obtain the spectral norm of matrices, but it can also be efficiently extended to find the spectral norm of the Jacobian of convolutional layers. This was done for example by [57], [65], using the fact that the transpose of a convolution operation (required to calculate Equation (3.4)) is a convolution as well, with a kernel that can be constructed from the original one by transposing the channel dimensions and flipping the spatial dimensions of the kernel. When the power method is used on a parameter matrix of a layer, we can make it even more efficient with a simple trick. We usually expect the parameter matrix to change only slightly during each training step, so we can store the result $\mathbf{u}_k$ during each training step, and start the power method with this vector as $\mathbf{u}_0$ during the following training step. With this trick, it is enough to do a single iteration of the power method at each training step. The power method is usually not differentiated through.

Fantasic Four proposed, in [62], allows upper bounding of the Lipschitz constant of a convolution. The given bound is generally not tight, so using the method directly does not give good results. Nevertheless, since various methods require a way of bounding the spectral norm to have convergence guarantees, Fantastic Four is often used.

3.1.2 Parameterized 1-Lipschitz Layers

This section provides an overview of the existing methods for providing 1-Lipschitz layers. This section describes methods in the literature that construct 1-Lipschitz convolutions: BCOP, Cayley, SOC, AOL, LOT, CPL, SLL, ONI, ECO, Sandwich. Nevertheless, some of them have not been compared, i.e., ECO, Sandwich, ONI proposed in [66]–[68] respectively. The reasons why they were not included in the main comparison are discussed in each dedicated paragraph.

BCOP *Block Convolution Orthogonal Parameterization (BCOP)* was introduced by Li *et al.* in [55] to extend a previous work by Xiao *et al.* [69] that focused on the importance of orthogonal initialization of the weights. For a $k \times k$ convolution, BCOP uses a set of $(2k-1)$ parameter matrices. Each of these matrices is orthogonalized using the algorithm by Bjorck & Bowie [63] (see also Section 3.1.1). Then, a $k \times k$ kernel is constructed from those matrices to guarantee that the resulting layer is orthogonal.

Cayley Another family of orthogonal convolutional and fully connected layers has been proposed by Trockman and Kolter [56] by leveraging the *Cayley Transform* [70], which maps a skew-symmetric matrix A into an orthogonal matrix Q using the relation

$$Q = (I - A)(I + A)^{-1}. \tag{3.6}$$

The transformation can be used to parameterize orthogonal weight matrices for linear layers in a straightforward way. For convolutions, the authors make use of the fact that circular padded convolutions are vector-matrix products in the Fourier domain. As long as all those vector-matrix products have orthogonal matrices, the full convolution will have an orthogonal Jacobian. For *Cayley Convolutions*, those matrices are orthogonalized using the Cayley transform.

SOC *Skew Orthogonal Convolution (SOC)* is an orthogonal convolutional layer presented by Singla *et al.* [59], obtained by leveraging the exponential convolution [71]. Analogously to the matrix case, given a kernel $L \in \mathbb{R}^{c\times c\times k\times k}$, the exponential convolution can be defined as

$$\exp(L)(x) := x + \frac{L \star x}{1} + \frac{L \star^2 x}{2!} + \cdots + \frac{L \star^k x}{k!} + \cdots , \tag{3.7}$$

where $\star^k$ denotes a convolution applied k-times. The authors proved that any exponential convolution has an orthogonal Jacobian matrix as long as L is skew-symmetric, providing a way of parameterizing 1-Lipschitz layers. In their work, the sum of the infinite series is approximated by computing only the first 5 terms during training and the first 12 terms during the inference, and L is normalized to have unitary spectral norm following the method presented in [62] (see Section 3.1.1).

AOL Prach and Lampert [54] introduced *Almost Orthogonal Layer (AOL)* layers. For any matrix P, they defined a diagonal rescaling matrix D with

$$D_{ii} = \left(\sum_j \left| P^\top P \right|_{ij} \right)^{-1/2} \tag{3.8}$$

and proved that the spectral norm of PD is bounded by 1. This result was used to show that the linear layer given by $l(x) = PDx+b$ (where P is the learnable matrix and D is given by eq. (3.8)) is 1-Lipschitz. Furthermore, the authors extended the idea so that it can also be efficiently applied to convolutions. This is done by calculating the rescaling in Equation (3.8) with the Jacobian J of a convolution instead of P. In order to evaluate it efficiently the authors express the elements of $J^\top J$ explicitly in terms of the kernel values.

LOT The layer presented by Xu *et al.* [60] extends the idea of [68] to use the *Inverse Square Root* of a matrix in order to orthogonalize it. Indeed, for any matrix V, the matrix $Q = V(V^T V)^{-\frac{1}{2}}$ is orthogonal. Similarly to the *Cayley* method, for the *layer-wise orthogonal training (LOT)* the convolution is applied in the Fourier frequency domain. To find the inverse square root, the authors rely on an iterative *Newton Method*. In details, defining $Y_0 = V^T V$, $Z_0 = I$, and

$$Y_{i+1} = \frac{1}{2} Y_i \left(3I - Z_i Y_i\right), \; Z_{i+1} = \frac{1}{2} \left(3I - Z_i Y_i\right) Z_i, \tag{3.9}$$

it can be shown that Y_i converges to $(V^TV)^{-\frac{1}{2}}$. In their proposed layer, the authors apply 10 iterations of the method for both training and evaluation.

CPL Meunier *et al.* [64] proposed the *Convex Potential Layer.* Given a non-decreasing 1-Lipschitz function σ (usually ReLU), the layer is constructed as

$$l(x) = x - \frac{2}{\|W\|_2^2} W^\top \sigma(Wx + b), \tag{3.10}$$

which is 1-Lipschitz by design. The spectral norm required to calculate $l(x)$ is approximated using the Power Method (see Section 3.1.1).

SLL The *SDP-based Lipschitz Layers (SLL)* proposed by Araujo *et al.* [72] combine the CPL layer with the upper bound on the spectral norm from AOL. The layer can be written as

$$l(x) = x - 2WQ^{-2}D^2\sigma\left(W^\top x + b\right), \tag{3.11}$$

where Q is a learnable diagonal matrix with positive entries and D is deduced by applying Equation (3.8) to $P = WQ^{-1}$.

Remark 1. Both CPL and SLL are non-linear by construction, so they can be used to construct a network without any further use of activation functions. However, carrying out some preliminary experiments, we empirically found that alternating CPL (and SLL) layers with MaxMin activation layers allows for achieving a better performance.

3.1.3 1-Lipschitz excluded from the comparison

In this section an overview of three methods in the literature is given, [66]–[68].

ONI The method proposed by Huang*et al.*, *Orthogonalization using Newton's Iteration (ONI)*, [68] relies on an orthogonalization strategy similar to the procedure used in LOT. Parameterization of orthogonal matrices is done by considering $(VV^\top)^{-\frac{1}{2}}V$, where the inverse square root is obtained through Newton's iterations. However, the extension to convolutions only uses a simple unrolling and does not provide a tight bound of the Lipschitz constant of the layer. Therefore, we did not provide a comparison of this method.

ECO *Explicitly Constructed Orthogonal (ECO)* convolutions [66] also do use properties of the Fourier domain in order to parameterize a convolution. However, they do not actually calculate the convolution in the Fourier domain, but instead parameterize a layer in the Fourier domain, and then use an inverse Fourier transformation to obtain a kernel from this parameterization. Nevertheless, we omitted the layer from the comparison since we noticed that the implementation provided by the authors does not produce 1-Lipschitz layers with our architecture.

Sandwich *Sandwich* layer is introduced by Wang and Manchester, [67], and is defined as follows

$$l(x) = \sqrt{2}A^T\Psi \,\mathrm{ReLU}\left(\sqrt{2}\Psi^{-1}Bx + b\right). \tag{3.12}$$

The authors propose a (simultaneous) parameterization of A and B, based on the Cayley Transform, that guarantees the whole layer to be 1-Lipschitz. They also extend the idea to convolutions. However, for this, they are required to apply two Fourier transformations as well as two inverse ones. During the training of Sandwich we faced some numerical errors. To investigate such errors, we tested a lighter version of the method — without the learnable rescaling Ψ — for the reason described in Remark 2, which shows that the rescaling Ψ inside the layer can be embedded into the bias term and hence the product $\Psi\Psi^{-1}$ can be omitted. Furthermore, during the training of the models within the Sandwich layers, a severe vanishing gradient phenomenon happens. Therefore, the layer is not compared with the other tested methods.

Remark 2. The learnable parameter Ψ of the sandwich layer corresponds to a scaling of the bias. In details, for each parameters A, B, b and $\Psi = \mathrm{diag}\left(e^{d_i}\right)$ there exists a rescaling of the bias $\tilde{b}$ such that

$$l(x) = \sqrt{2}A^T\Psi\mathrm{ReLU}\left(\sqrt{2}\Psi^{-1}Bx + b\right) = \sqrt{2}A^T\mathrm{ReLU}\left(\sqrt{2}Bx + \tilde{b}\right) \tag{3.13}$$

Proof. Observing that for each $\alpha > 0$ and $x \in \mathbb{R}$, $\mathrm{ReLU}\left(\alpha x\right) = \alpha\mathrm{ReLU}\left(x\right)$, and that

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad \Psi^{-1}\mathbf{x} = \begin{bmatrix} e^{-d_1}x_1 \\ \vdots \\ e^{-d_n}x_n \end{bmatrix},$$

the following identity holds

$$\begin{aligned} l(x) &= \sqrt{2}A^T\Psi\mathrm{ReLU}\left(\sqrt{2}\Psi^{-1}Bx + b\right) \\ &= \sqrt{2}A^T\Psi\mathrm{ReLU}\left(\sqrt{2}\Psi^{-1}\left(\sqrt{2}Bx + \Psi b\right)\right) \\ &= \sqrt{2}A^T\Psi\Psi^{-1}\mathrm{ReLU}\left(\sqrt{2}Bx + \Psi b\right) \\ &= \sqrt{2}A^T\mathrm{ReLU}\left(\sqrt{2}Bx + \Psi b\right). \end{aligned} \tag{3.14}$$

Considering $\tilde{b} = \Psi b$ concludes the proof. □

3.2 Theoretical Comparison

As illustrated in the last section, various ideas and methods have been proposed to parameterize 1-Lipschitz layers. This causes the different methods to have very different properties and requirements. This section aims at highlighting the properties of the different algorithms, focusing on the algorithmic complexity and the required memory.

Table 3.1 provides an overview of the computational complexity and memory requirements for the different layers considered in the previous section. For the sake of clarity, the analysis is performed by considering separately the transformations applied to the input of the layers and those applied to the weights to ensure the 1-Lipschitz constraint. Each of the two sides of the table contains three columns: i) *Operations* contains the most costly transformations applied to the input as well as to the parameters of different layers; ii) *MACS* reports the computational complexity expressed in multiply-accumulate operations (MACS) involved in the transformations (only leading terms are presented); iii) *Memory* reports the memory required by the transformation during the training phase.

Method	Input Transformations			Parameter Transformations		
	Operations	MACS $\mathcal{O}(\cdot)$	Memory	Operations	MACS $\mathcal{O}(\cdot)$	Memory $\mathcal{O}(\cdot)$
Standard	CONV	C	M	-	-	P
AOL	CONV	C	M	CONV	c^3k^4	$5P$
BCOP	CONV	C	M	BnB & MMs	$c^3kt + c^3k^3$	$c^2kt + c^2k^3$
Cayley	FFTs & MVs	bs^2c^2	$\frac{5}{2}M$	FFTs & INVs	s^2c^3	$\frac{3}{2}s^2c^2$
CPL	CONVs & ACT	$2C$	$3M$	power method	$s^2c^2k^2$	$P + s^2c$
LOT	FFTs & MVs	bs^2c^2	$3M$	FFTs & MMs	$4s^2c^3t$	$4s^2c^2t$
SLL	CONVs & ACT	$2C$	$3M$	CONVs	c^3k^4	$5P$
SOC	CONVs	Ct_1	Mt_1	F4	$c^2k^2t_2$	P

Table 3.1: **Computational complexity and memory requirements of different methods.** We report multiply-accumulate operations (MACS) as well as memory requirements (per layer) for batch size b, image size $s \times s \times c$, kernel size $k \times k$ and number of inner iterations t. We use $C = bs^2c^2k^2$, $M = bs^2c$ and $P = c^2k^2$. For a detailed explanation of what is reported see Section 3.2.

At training time, both input and weight transformations are required, thus the training complexity of the forward pass can be computed as the sum of the two corresponding MACS columns of the table. Similarly, the training memory requirements can be computed as the sum of the two corresponding Memory columns of the table. For the considered operations, the cost of the backward pass during training has the same computational complexity as the forward pass, and therefore increases the overall complexity by a constant factor. At inference time, all the parameter transformations can be computed just once and cached afterward. Therefore, the inference complexity is equal to the complexity due to the input transformation (column 3 in the table). The memory requirements at inference time are much lower than those needed at the training time since intermediate activation values do not need to be stored in memory, hence we do not report them in Table 3.1.

Note that all the terms reported in Table 3.1 depend on the batch size b, the input size $s \times s \times c$, the number of inner iterations of a method t, and the kernel size $k \times k$. (Often, t is different at training and inference time.) For the sake of clarity, the MACS of a naive convolution implementation is denoted by C ($C = bs^2c^2k^2$), the number of inputs of a layer is denoted by M ($M = bs^2c$), and the size of the kernel of a standard convolution is denoted by P ($P = c^2k^2$). Only the leading terms of the computations are reported in Table 3.1. In order to simplify some terms, we assume that $c > \log_2(s)$ and that rescaling a tensor (by a scalar) as well as adding two tensors does not require any memory in order to do backpropagation. We also assume that each additional activation does require extra memory. All these assumptions have been verified to hold within *PyTorch*, [73]. Also, when the algorithm described in the paper and the version provided in the supplied code differed, we considered the algorithm implemented in the code.

The transformations reported in the table are convolutions (CONV), Fast Fourier Transformations (FFT), matrix-vector multiplications (MV), matrix-matrix multiplications (MM), matrix inversions (INV), as well as applications of an activation function (ACT). The application of algorithms such as Bjorck & Bowie (BnB), power method, and Fantastic 4 (F4) is also reported (see Section 3.1.1 for descriptions).

3.2.1 Analysis of the computational complexity

It is worth noting that the complexity of the input transformations (in Table 3.1) is similar for all methods. This implies that a similar scaling behavior is expected at inference time for the models. Cayley and LOT apply an FFT-based convolution and have computational complexity independent of the kernel size. CPL and SLL require two convolutions, which makes them slightly more expensive at inference time. Notably, SOC requires multiple convolutions, making this method more expensive at inference time.

At training time, parameter transformations need to be applied in addition to the input transformations during every forward pass. For SOC and CPL, the input transformations always dominate the parameter transformations in terms of computational complexity. This means the complexity scales like c^2, just like a regular convolution, with a further factor of 2 and 5 respectively. All other methods require parameter transformations that scale like c^3, making them more expensive for larger architectures. In particular, we do expect Cayley and LOT to require long training times for larger models since the complexity of their parameter transformations further depends on the input size.

3.2.2 Analysis of the training memory requirements

The memory requirements of the different layers are important since they determine the maximum batch size and the type of models we can train on a particular infrastructure. At training time, typically all intermediate results are kept in memory to perform backpropagation. This includes intermediate results for both input and parameter transformations. The input transformation usually preserves the size, and therefore the memory required is usually of $\mathcal{O}(M)$. Therefore, for the input transformations, all methods require memory not more than a constant factor worse than standard convolutions, with the worst method being SOC, with a constant t_1, typically equal to 5.

In addition to the input transformation, we also need to store the intermediate results of the parameter transformations in memory to evaluate the gradients. Again, most methods approximately preserve the sizes during the parameter transformations, and therefore the memory required is usually of order $\mathcal{O}(P)$. Exceptions to this rule are Cayley and LOT, which contain a much larger $\mathcal{O}(s^2c^2)$ term, as well as BCOP.

3.3 Methodology

This section presents the methodology adopted for comparing the performance of the considered layers concerning different metrics. To have a fair and meaningful comparison among the various models, all the proposed layers have been evaluated using the same architecture, loss function, and optimizer. Since, according to the data reported in Table 3.1, different layers may have different throughput, to have a fair comparison for the tested metrics, we limited the total training time instead of fixing the number of training epochs. Results are reported for training times of 2h, 10h, and 24h on one A100 GPU.

3.3.1 Architecture

Our architecture is a standard convolutional network that doubles the number of channels whenever the resolution is reduced [18], [56]. For each method, we tested architectures of different sizes. We denoted them as XS, S, M and L, depending on the number of parameters, according to the criteria in Table 3.4, ranging from 1.5M to 100M parameters. The architecture used for the experiments is summarized in Tables 3.2 and 3.3. It is a standard convolutional architecture, that doubles the number of channels whenever the resolution is reduced.

Layer name	Output size
Input	$32 \times 32 \times 3$
Zero Channel Padding	$32 \times 32 \times w$
Conv (1×1 kernel size)	$32 \times 32 \times w$
Activation	$32 \times 32 \times w$
Downsize Block(k)	$16 \times 16 \times 2w$
Downsize Block(k)	$8 \times 8 \times 4w$
Downsize Block(k)	$4 \times 4 \times 8w$
Downsize Block(k)	$2 \times 2 \times 16w$
Downsize Block(1)	$1 \times 1 \times 32w$
Flatten	$32w$
Linear	$32w$
First Channels(c)	c

Table 3.2: **Architecture.** It depends on width parameter w, kernel size k ($k \in \{1, 3\}$) and the number of classes c. For details of the *Downsize Block* see table 3.3.

Note that we exclusively use convolutions with the same input and output size as an attempt to make the model less dependent on the initialization used by the convolutional layers. We use kernel size 3 in all our main experiments. The layer *Zero Channel Padding* in Table 3.2 just appends channels with value

	Layer name	Kernel size	Output size
$5 \times \{$	Conv	$k \times k$	$s \times s \times t$
	Activation	-	$s \times s \times t$
	First Channels	-	$s \times s \times t/2$
	Pixel Unshuffle	-	$s/2 \times s/2 \times 2t$

Table 3.3: Downsize Block(k) with input size $s \times s \times t$:

Size	Parameters (millions)	w
XS	$1 < p < 2$	16
S	$4 < p < 8$	32
M	$16 < p < 32$	64
L	$64 < p < 128$	128

Table 3.4: Number of parameters for different model sizes, as well as the *width parameter* w such that the architecture in table 3.2 has the correct size.

0 to the input, and the layer *First Channels(c)* outputs only the first c channels, and ignores the rest. Finally, the layer *Pixel Unshuffle* (implemented in *PyTorch*) takes each $2 \times 2 \times c$ patches of an image and reshapes them into size $1 \times 1 \times 4c$. For each 1-Lipschitz layer, we also test architectures of different sizes. In particular, we define 4 categories of models based on the number of parameters. We call those categories XS, S, M and L. See Table 3.4 for the exact numbers. In this table we also report the *width parameter* w that ensures our architecture has the correct number of parameters.

Remark 3. For most methods, the number of parameters per layer are about the same. There are two exceptions, BCOP and Sandwich. BCOP parameterizes the convolution kernel with c input channels and c output channels using a matrix of size $c \times c$ and $2(k-1)$ matrices of size $c \times c/2$. Therefore, the number of parameters of a convolution using BCOP is kc^2, less than the k^2c^2 parameters of a plain convolution. The Sandwich layer has about twice as many parameters as the other layers for the same width, as it parameterizes two weight matrices, A and B in Equation (3.12), per layer.

3.3.2 Training

Since one of the main goals is to evaluate what is the best model to use given a certain time budget, we measure the time per epoch as described in Section 3.3.5 on an A100 GPU with 80GB memory for different methods and different model sizes. Then we estimate the number of epochs we can do in our chosen time budget of either 2h, 10h or 24h, and use that many epochs to train our models. The amount of epochs corresponding to the given time budget is summarized in Table 3.5. Being the goal metric is certified robust accuracy for perturbation of

	CIFAR-10&100				TinyImageNet			
	XS	S	M	L	XS	S	M	L
AOL	837	763	367	83	223	213	123	34
BCOP	127	125	94	24	50	50	39	11
CPL	836	797	522	194	240	194	148	63
Cayley	356	214	70	17	138	86	30	8
ECO	399	387	290	162	142	131	95	54
LOT	222	68	11	-	83	29	5	-
SLL	735	703	353	79	242	194	118	32
SOC	371	336	201	77	122	87	63	27
Param.s (M)†	1.57	6.28	25.12	100.46	1.58	6.29	25.16	100.63

† BCOP has less parameters overall, see Remark 3.

Table 3.5: Budget of training epochs for different model sizes, layer types and datasets. Batch size and training time are set to be 256 and 2h respectively for all the architectures.

maximal size $\epsilon = 36/255$, we use the loss function proposed by [54], where the margin parameter is set to $2\sqrt{2}\epsilon$, and the temperature parameter is set to $t = 1/4$. This allows aiming at the robustness of maximal size 2ϵ during training. We use SGD with a momentum of 0.9 for all experiments.We considered the learning rate scheduler *OneCycleLR*, as described by [74], with default values as in *PyTorch*. We set the batch size to 256 for all experiments.

3.3.3 Random search of hyperparameter

Since different methods benefit from different learning rates and weight decay, for each setting (model size, method and dataset), we used the best values resulting from a random search performed on multiple training runs on a validation set composed of 10% of the original training set. More specifically, we did 16 runs with learning-rate of the form 10^x, where x is sampled uniformly in the interval $[-4, -1]$, and with a weight decay of the form 10^x, where x is sampled uniformly in the interval $[-5.5, -3.5]$. Finally, we selected the learning rate and weight decay corresponding to the run with the highest validation certified robust accuracy for radius 36/255. We use these hyperparameters found also for the experiments with longer training time. Each run is performed by training the setting (model size, method and dataset) and the sampled learning rate and weight decay for $2h$. The number of epochs is summarized in Table 3.5.

3.3.4 Datasets

We evaluate on three different datasets, CIFAR-10, CIFAR-100 [41] and Tiny ImageNet [75]. For CIFAR-10 and CIFAR-100 we use the architecture described in Table 3.2. Since the architectures are identical, so are time- and memory requirements, and therefore also the epoch budget. As preprocessing we subtract the dataset channel means from each image. As data augmentation at training time we apply random crops (4 pixels) and random flipping. In order to assess the behavior on larger images, we replicate the evaluation on the Tiny ImageNet dataset [75]: a subset of 200 classes of the ImageNet [19] dataset, with images scaled to have size 64×64. In order to allow for the larger input size of this dataset, we add one additional *Downsize Block* to our model. We also divide the width parameter (given in Table 3.4) by 2 to keep the amount of parameters similar. We again subtract the channel mean for each image. As data augmentation we we us *RandAugment* [76] with 2 transformations of magnitude 9 (out of 31).

3.3.5 Metrics

All the considered models were evaluated based on three main metrics: the *throughput*, the required memory, and the certified robust accuracy.

Throughput and epoch time The *throughput* of a model is the average number of samples that the model can process per second. It determines how many epochs are processed in a given time frame. The evaluation of the *throughput* is performed on an `NVIDIA A100 80GB PCIe GPU` [*]. based on the average time of 100 mini-batches. We measured the inference throughput with cached parameter transformations.

Memory required Layers that require less memory allow for larger batch size, and the memory requirements also determine the type of hardware we can train a model on. For each model, we measured and reported the maximal GPU memory occupied by tensors using the function `max_memory_allocated` provided by the *PyTorch* framework. This is not exactly equal to the overall GPU memory requirement but gives a fairly good approximation of it. Note that the model memory measured in this way also includes additional memory required by the optimizer (e.g. to store the momentum term) as well as by the activation layers in the forward pass. However, this additional memory should be at most of order $\mathcal{O}(M + P)$. As for the throughput, we evaluated and cached all calculations independent of the input at inference time.

Certified robust accuracy In order to evaluate the performance of a 1-Lipschitz network, the standard metric is the *certified robust accuracy*. An input is classified certifiably robustly with radius ϵ by a model, if no perturbations of the input with norm bounded by ϵ can change the prediction of the model. Certified robust accuracy measures the proportion of examples that are classified correctly as well as certifiably robustly. For 1-Lipschitz models, a lower bound of the certified ϵ-robust accuracy is the ratio of correctly classified inputs such that $\mathcal{M}_f(x_i, l_i) > \epsilon\sqrt{2}$ where the *margin* $\mathcal{M}_f(x, l)$ of a model f at input x with label l, given as $\mathcal{M}_f(x, l) = f(x)_l - \max_{j \neq l} f_j(x)$, is the difference between target class score and the highest score of a different class. For details, see [77].

3.4 Experimental Results

This section presents the results of the comparison performed by applying the methodology discussed in Section 3.3. The results related to the different metrics

[*]NIVIDA A100 80GB PCIe GPU product brief

are discussed in dedicated subsections and the key takeaways are summarized in the radar-plot illustrated in Figure 3.1.

3.4.1 Training and inference times

Figure 3.2 plots the training time per epoch of the different models as a function of their size, while Figure 3.3 plots the corresponding inference throughput for the various sizes as described in Section 3.3. As described in Table 3.2, the model base width, referred to as w, is doubled from one model size to the next. We expect the training and inference time to scale with w similarly to how individual layers scale with their number of channels, c (in Table 3.1). This is because the width of each of the 5 blocks of our architecture is a constant multiple of the base width, w.

The training time increases (at most) about linearly with w for standard convolutions, whereas the computational complexity of each single convolution scales like c^2. This suggests that parallelism on the GPU and the overhead from other operations (activations, parameter updates, etc.) are important factors determining the training time. This also explains why CPL (doing two convolutions,

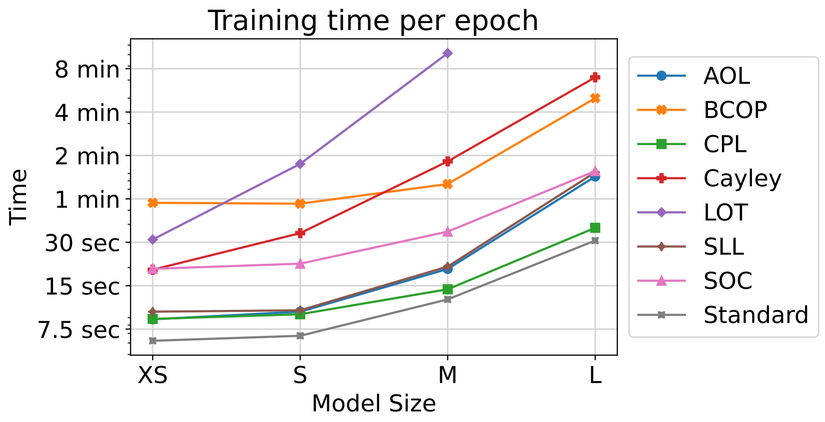


Figure 3.2: Training time per epoch (on CIFAR-10) for different methods and different model sizes.

with identical kernel parameters) is only slightly slower than a standard convolution, and SOC (doing 5 convolutions) is only about 3 times slower than the standard convolution. The AOL and SLL methods also require times comparable to a standard convolution for small models, although eventually, the c^3 term in the computation of the rescaling makes them slower for larger models. Finally, Cayley, LOT, and BCOP methods take much longer training times per epoch. For Cayley and LOT this behavior was expected, as they have a large $\mathcal{O}(s^2c^3)$ term in their computational complexity. See Table 3.1 for further details.

At inference time transformations of the weights are cached, therefore some methods (AOL, BCOP) do not have any overhead compared to a standard convolution. As expected, other methods (CPL, SLL, and SOC) that apply additional

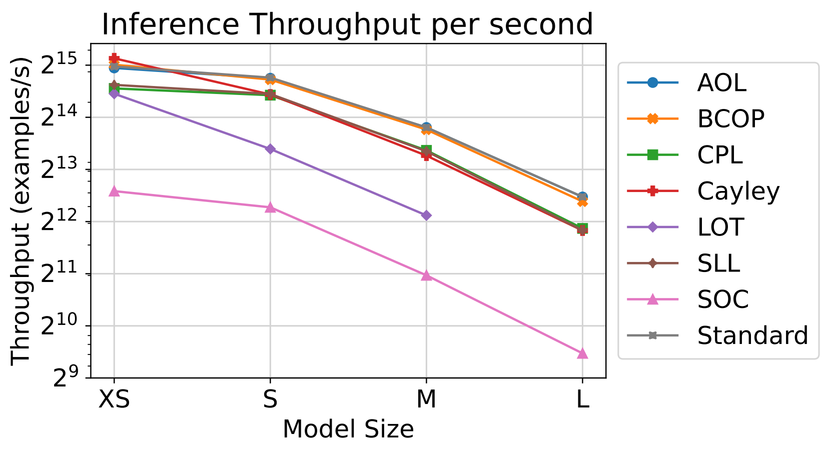


Figure 3.3: Inference throughput for different methods as a function of their size for CIFAR-10 sizes input images. All parameter transformations have been evaluated and cached beforehand

convolutions to the input suffer from a corresponding overhead. Finally, Cayley and LOT have a slightly different throughput due to their FFT-based convolution. Among them, Cayley is about twice as fast because it involves a real-valued FFT rather than a complex-valued one. From Figure 3.3, it can be noted that cached Cayley and CPL have the same inference time, even though CPL uses twice the number of convolutions. We believe this is due to the fact that the conventional FFT-based convolution is quite efficient for large filters, but PyTorch implements a faster algorithm, *i.e.*, *Winograd*, [78], that can be up to 2.5 times faster.

3.4.2 Training memory requirements

The training and inference memory requirements of the various models (measured as described in Section 3.3.5) are reported in Figure 3.4 as a function of the model size. The results of the theoretical analysis reported in Table 3.1 suggest that the training memory requirements always have a term linear in the number of channels, c (usually the activations from the forward pass), as well as a term quadratic in c (usually the weights and all transformations applied to the weights during the forward pass). This behavior can also be observed from Figure 3.4. For some of the models, the memory required approximately doubles from one model size to the next one, just like the width. This means that the linear term dominates (for those sizes), which makes those models relatively cheap to scale up.

For the BCOP, LOT, and Cayley methods, the larger coefficients in the c^2 term (for LOT and Cayley the coefficient is even dependent on the input size, s^2) cause this term to dominate. This makes it much harder to scale those methods to more parameters. Method LOT requires huge amounts of memory, in particular LOT-L is too large to fit in 80GB GPU memory. Note that at test time, the memory requirements are much lower, because the intermediate activation values do not need to be stored, as there is no backward pass. Therefore, at inference time, most methods require a very similar amount of memory as a standard convolution. The Cayley and LOT methods require more memory since perform the calculation

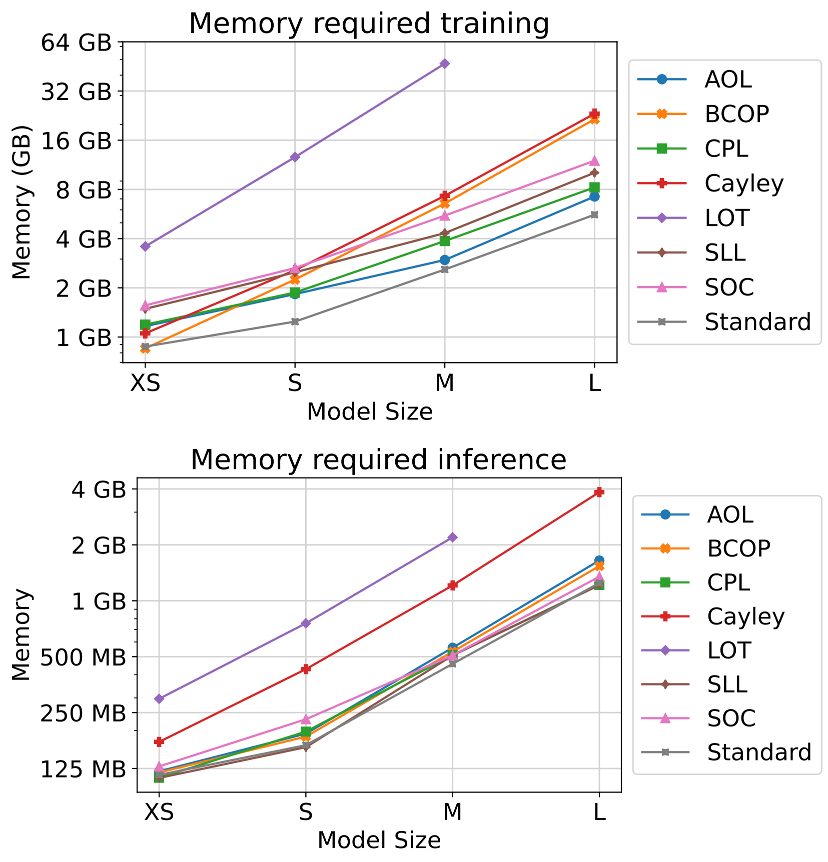


Figure 3.4: Memory required at training and inference time for input size 32×32.

in the Fourier space, as they create an intermediate representation of the weight matrices of size $\mathcal{O}(s^2c^2)$.

3.4.3 Certified robust accuracy

The results related to the accuracy and the certified robust accuracy for the different methods, model sizes, and datasets measured on a 24h training budget are summarized in Table 3.6. The differences among the various model sizes are also highlighted in Figure 3.5 by reporting the sorted values of the certified robust accuracy. The reader can compare our results with the state-of-the-art certified robust accuracy summarized in Table 3.7. However, it is worth noting that, to reach state-of-the-art performance, authors often carry out experiments using large model sizes and long training times, which makes it hard to compare the methods themselves. On the other hand, the evaluation proposed in this paper allows a fairer comparison among the different methods, since it also considers timing and memory aspects. This restriction based on time, rather than the number of epochs, ensures that merely enlarging the model size does not lead to improved performance, as bigger models typically process fewer epochs of data. Indeed, in our results in Figure 3.5 it is usually the M (and not the L) model that performs best. To assign a score that combines the performance of the methods over all the three datasets, we sum the number of times that each method is ranked in

	Accuracy [%]				Robust Accuracy [%]			
Methods	XS	S	M	L	XS	S	M	L
CIFAR-10								
AOL	71.7	73.6	73.4	73.7	59.1	60.8	61.0	**61.5**
BCOP	71.7	73.1	74.0	74.6	58.5	59.3	60.5	**61.5**
CPL	74.9	76.1	76.6	76.8	62.5	64.2	65.1	**65.2**
Cayley	73.1	74.2	74.4	73.6	59.5	**61.1**	61.0	60.1
LOT	75.5	76.6	72.0	-	63.4	**64.6**	58.7	-
SLL	73.7	74.2	75.3	74.3	61.0	62.0	**62.8**	62.3
SOC	74.1	75.0	76.9	76.9	61.3	62.9	**66.3**	65.4
CIFAR-100								
AOL	40.3	43.4	44.3	41.9	27.9	31.0	**31.4**	29.7
BCOP	41.4	42.8	43.7	42.2	28.4	30.1	**31.2**	29.2
CPL	42.3	-	45.2	44.3	30.1	-	**33.2**	32.1
Cayley	42.3	43.9	43.5	42.9	29.2	**30.5**	30.5	29.5
LOT	43.5	45.2	42.8	-	30.8	**32.5**	29.6	-
SLL	41.4	42.8	42.4	42.1	28.9	**30.5**	29.9	29.6
SOC	43.1	45.2	47.3	46.2	30.6	32.6	**34.9**	33.5
Tiny ImageNet								
AOL	26.6	29.3	30.3	30.0	18.1	19.7	**21.0**	20.6
BCOP	22.4	26.2	27.6	27.0	13.8	16.9	**17.2**	16.8
CPL	28.3	29.3	29.8	30.3	18.9	19.7	**20.3**	20.1
Cayley	27.8	29.6	30.1	27.2	17.9	**19.5**	19.3	16.7
LOT	30.7	32.5	28.8	-	20.8	**21.9**	18.1	-
SLL	25.1	27.0	26.5	27.9	16.6	18.4	17.7	**18.8**
SOC	28.9	28.8	32.1	32.1	18.9	18.8	**21.2**	21.1

Table 3.6: Certified robust accuracy for radius $\epsilon = 36/255$ on the evaluated datasets. Training is performed for 24 hours.

the first position, in the top-3, and top-10 positions. In this way, top-1 methods are counted three times, and top-3 methods are counted twice. The scores in the radar-plot shown in Figure 3.1 are based on those values.

Among all methods, SOC achieved a top-1 robust accuracy twice and a top-3 one 6 times, outperforming all the other methods. CPL ranks twice in the top-3 and 9 times in the top-10 positions, showing that it generally has a more stable performance compared with other methods. LOT achieved the best certified robust accuracy on Tiny ImageNet, appearing further 5 times in the top-10. AOL did not perform very well on CIFAR-10, but reached more competitive results on Tiny ImageNet, ending up in the top-10 a total of 5. An opposite effect can be observed for SLL, which performed reasonably well on CIFAR-10, but not so well on the two datasets with more classes, placing in the top-10 only once. This result is tied with BCOP, which also has only one model in the top-10. Finally, Cayley

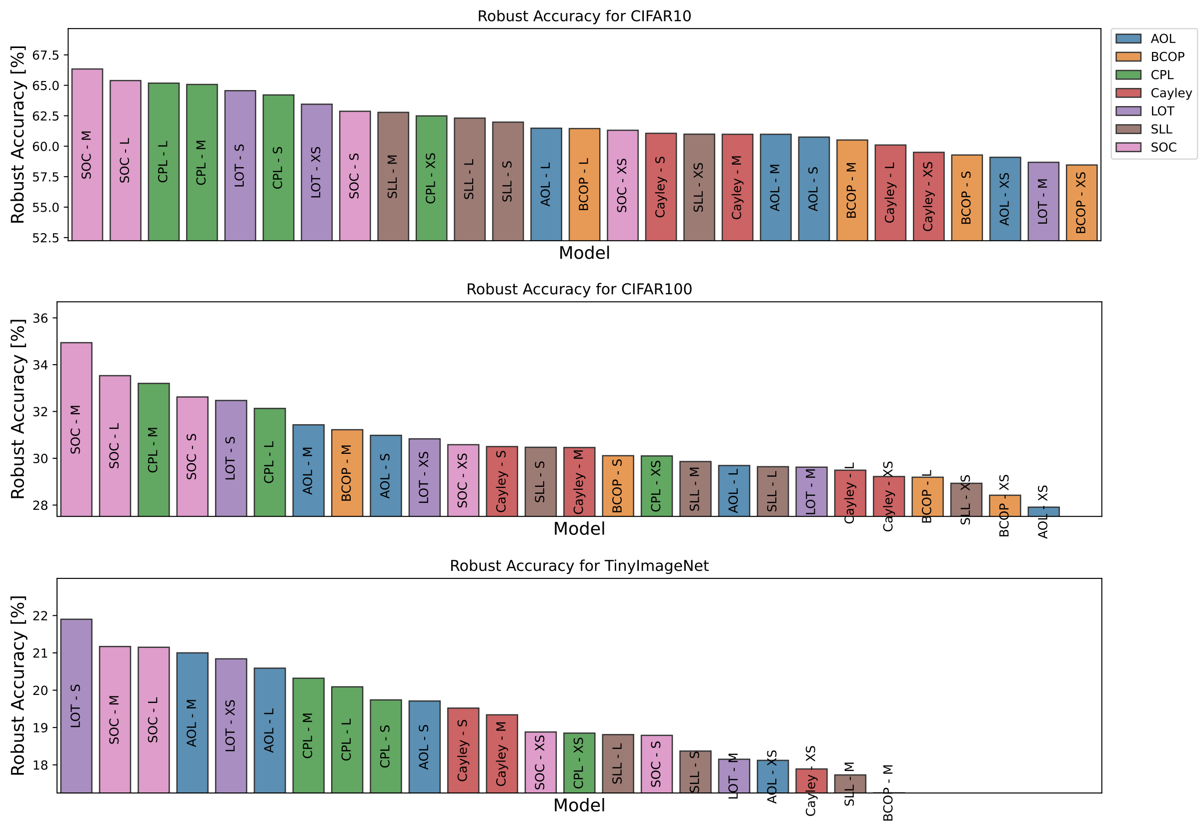


Figure 3.5: Certified robust accuracy by decreasing order. Note that the axes do not start at 0.

Method	Std.Acc [%]	Certifiable Accuracy [%] $\varepsilon = \frac{36}{255}$	Certifiable Accuracy [%] $\varepsilon = \frac{72}{255}$	Parameters
BCOP Large [55]	72.1	58.2	-	2M
Cayley KW-Large [56]	75.3	59.1	-	2M
SOC LipNet-25 [59]	76.4	61.9	-	24M
AOL Large [54]	71.6	64.0	56.4	136M
LOT LipNet-25 [60]	76.8	64.4	49.8	27M
SOC LipNet-15 + CRC [79]	79.4	67.0	52.6	21M
CPL XL [64, Table1]	78.5	64.4	48.0	236M
SLL X-Large [72]	73.3	65.8	58.4	236M

Table 3.7: **SOTA from the literature on CIFAR-10** sorted by publication date (from older to newer). Readers can note that there is a clear trend of increasing the model dimension to achieve higher robust accuracy.

is consistently outperformed by the other methods. The very same analysis can be applied to the clean accuracy, where the main difference is that Cayley performs slightly better for that metric. Furthermore, it is worth highlighting that CPL is sensitive to weight initialization. We faced numerical errors during the 10h and 24h training of the small model on CIFAR-100.

Chapter 4

Signed Distance Classifiers

As introduced in the previous chapters, In the last few years, a large number of methods for crafting adversarial examples have been presented [5], [6], [9], [16]. In particular, [7], [9] proposed methods to find the minimal adversarial perturbation (MAP) or, equivalently, the closest adversarial example for a given input x.

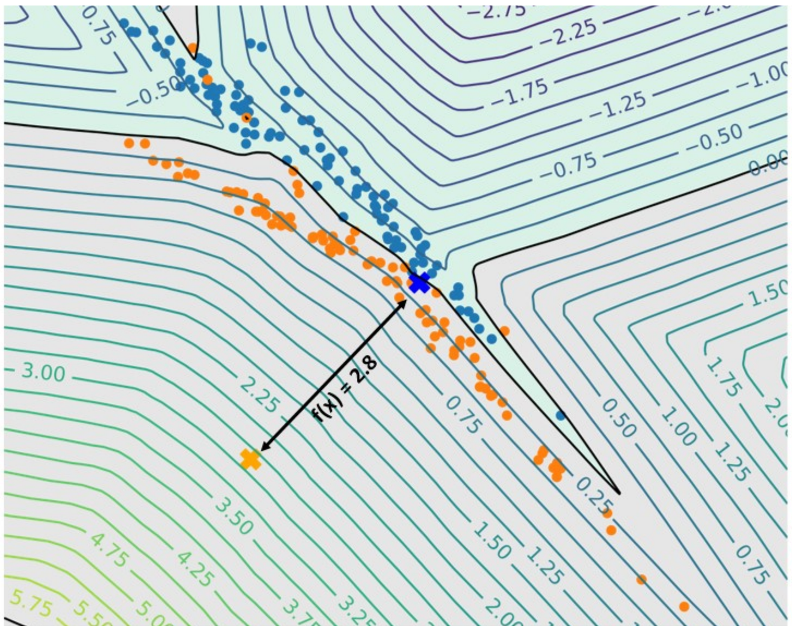


Figure 4.1: An example of a binary SDC. Observe that the countour lines are parallel-curves of the classification boundary (the black curve) and the output of the model in a x (the orange cross) directly provides the distance from the closest point in the classification boundary (the blue cross).

Such a perturbation directly provides the distance of x from the classification boundary, which, given a maximum magnitude of perturbation, can be used to verify the trustworthiness of the prediction [36] and design robust classifiers [50], [80]. Note that, when the MAP is known, one can check on-line whether a certain input x can be perturbed with a bounded-magnitude perturbation to change the classification result. If this is the case, the network itself can signal the unsafeness of the result. Unfortunately, due the hard complexity of the algorithms for solving the MAP problem on classic models, the aforementioned strategies are not suited for efficiently certifying the robustness of classifiers [14]. To achieve provable guarantees, other works focused on designing network models with bounded Lipschitz constant that, by construction, offers a lower bound of the MAP as the network output. These particular models can be obtained by composing orthogonal layers [53], [55], [56], [59], [81] and norm-preserving activation functions, such as those presented by [58], [82]. However, despite the satisfaction of the Lipschitz inequality, these models do not provide the exact boundary distance but only a lower bound.

This chapter introduces a new family of classifiers, namely *Signed Distance Classifier (SDC)*, that *straighten the Lipschitz lower bound by outputting the exact distance of an input x from the classification boundary.* SDC can then solve the MAP problem as a result of the network inference (see Figure 4.1). From a theoretical point of view, we extend the characterization property of the signed distance functions to a multi-class classifier. From a practical perspective, we address such a theoretical model by presenting a new architecture, named Unitary-Gradient Neural Network (UGNN), *having unitary gradient (under the Euclidean norm) in each differentiable point.* In summary, this chapter provides the following contributions:

- It introduces a notable family of classifiers, named SDC, which provide as output the distance of an input x from the classification boundary.
- It provides a novel network architecture, named UGNN, which, to the best of our knowledge, represents the first practical approximation of an SDC.
- It shows that the abs function can replace other more expensive norm-preserving activation functions without introducing a significant accuracy loss. Furthermore, it proposes a new layer named *Unitary Pair Difference*, which is a generalization of a fully connected orthogonal layer.
- It assesses the performance, advantages, and limitations of the proposed architecture through a close comparison with the state-of-the-art models in the field of the Lipschitz-Bounded Neural Networks.

Structure: After a discussion of the related work, we provide a definition and a characterization of the SDCs by first introducing the concept for a binary classifier and then extending it to a multi-class classifier, where a characterization property is discussed. Then, we propose the UGNN to practically address the required properties of an SDC. Finally, we discuss a set of experiments to evaluate UGNNs and discuss future works.

4.1 Related works

The evaluation and the provable verification of the robustness of a classification model can be addressed by computing the MAP in a given point x [83]. Since that computation involves solving a minimum problem with non-linear constraints, the community focused on designing faster algorithms to provide an accurate estimation of the distance to the classification boundary [9]–[11]. However, all these algorithms require multiple forward and backward steps, and hence are not suited for an online application [14].

On the other side, since the sensitiveness to input perturbations strictly depends on the Lipschitz constant of the model, knowing the local Lipschitz constant in a neighborhood of x provides a lower bound of the MAP in x [84]. In formulas, for a L-Lipschitz neural network f, a lower bound of MAP is deduced by considering $\frac{1}{L\sqrt{2}}(f_l(x) - f_s(x))$, where l, s are the first and the second top-2 components, respectively. However, for common DNNs, obtaining a precise estimation of L is still computationally expensive [85], thus also this strategy is not suited for an online application. For these reasons, recently, other works focused on developing neural networks with a bounded Lipschitz constant, as thoroughly discussed in the Chapter 3 Similarly, [81] considered neural networks f in which each component f_i is 1-Lipschitz, thus, differently from the 1-Lipschitz networks mentioned before, given a sample x, the lower bound of MAP is deduced by $\frac{1}{2}(f_l(x) - f_s(x))$.

Other authors leveraged *orthogonal* weight matrices to pursue the same objective. For instance, [86] showed that a ReLU Multi-Layer Perceptron merely composed of orthogonal layers is 1-Lipschitz. Indeed, an orthogonal matrix W (i.e. such that $WW^T = I$ or $W^TW = I$) has a unitary spectral norm, $\|W\| = 1$. Roughly speaking, orthogonal fully connected and convolutional layers can be obtained by *Regularization* or *Parameterization*. The former methods include a regularization term in the training loss function to encourage the orthogonality of the layers, e.g., [53] use $\beta\|W^TW - Id\|^2$. The latter methods, instead, consider a parameterization of the weight $W(\theta)$ depending on an unconstrained parameter θ so that, for each θ, $W(\theta)$ is an orthogonal weight matrix [56], [58]. For convolutional layers, a regularization strategy can be applied, since they can be written as matrix-vector product through a structured matrix [87]. However, recent parameterized strategies as BCOP [55], CayleyConv [56], and Skew Convolution [59] come out as efficient and performant alternatives. Nevertheless, more details about these strategies are provided in the last chapter of the thesis.

Differently from the aforementioned methods, this chapter defines an SDC, as a function f that provides the MAP by computing $f_l(x) - f_s(x)$, thus tightening the lower bounds provided by L-Lipschitz classifiers. Furthermore, we present the UGNN, designed by properly leveraging the previous orthogonal parameterized strategies, as the first architecture that approximates a theoretical SDC.

4.2 Signed Distance Classifier

In this context, a classifier $\hat{k} : \mathcal{X} \to \mathcal{Y}$ maps the input domain into a finite set of labels $\mathcal{Y}$. The concept of *robustness* is formally stated in the following definition.

Definition 3 (robustness)**.** A classifier $\hat{k}$ is ε-robust in an input $x \in \mathbb{R}^n$ (or equivalently, a classification $\hat{k}(x)$ is ε-robust), if $\hat{k}(x + \delta) = \hat{k}(x)$ for any perturbation δ with $\|\delta\| < \varepsilon$, where $\|\cdot\|$ is the Euclidean norm.

4.2.1 The case of a binary classifier

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a binary classifier that provides a classification of an input x based on its sign, i.e., $\hat{k}(x) = \operatorname{sgn}(f(x))$, and let $\mathcal{B}_f := \{x \in \mathbb{R}^n : f(x) = 0\}$ be the classification boundary of f. Given an input sample x, the *MAP* problem for a binary classifier is defined as follows:

$$\begin{aligned} d_f(x) := \inf_{p \in \mathbb{R}^n} \quad & \|p - x\| \\ \text{s.t.} \quad & f(p) = 0, \end{aligned} \tag{4.1}$$

where d_f represents the distance function from the boundary $\mathcal{B}_f$. The *closest adversarial example* to x is defined as the unique x^* (if any) such that $d_f(x) = \|x - x^*\|$ and $\operatorname{sgn}(f(x)) \neq \operatorname{sgn}(f(x^*))$. Observe that Problem (4.1) is equivalent to the definition of *Minimal Adversarial Perturbation* in [6].

Certifiable robustness. We refer to $\delta^* = x^* - x$ as the *perturbation* that realizes the MAP. By definition of $d_f(x)$, for each perturbation δ with $\|\delta\| < d_f(x)$ it holds $\hat{k}(x + \delta) = \hat{k}(x)$; hence, $\hat{k}$ is certifiable $d_f(x)$-robust in x.

A *Signed Distance Function* d_f^* is defined as follows:

$$d_f^*(x) = \begin{cases} d_f(x) & x \in R_+ \\ -d_f(x) & x \notin R_+. \end{cases} \tag{4.2}$$

where $R_+ = \{f > 0\}$. Following this definition, a signed distance function d_f^* satisfies intriguing properties that make it highly interesting for *robustness evaluation, verification, and certifiable prediction*. In particular, d_f^* provides the same classification of f, since $\operatorname{sgn}(d_f^*(x)) = \operatorname{sgn}(f(x))$ for each $x \in \mathbb{R}^n$. Furthermore, the gradient $\nabla d_f^*(x)$ coincides with the direction of the shortest path to reach the closest adversarial example to x [88, Thm. 4.8].

Observation 1. Let $x \in \mathbb{R}^n$, if there exists a unique $x^* \in \mathcal{B}_f$ such that $d_f(x) = \|x - x^*\|$, then d_f^* is differentiable in x such that

$$\nabla d_f(x) = \frac{x - x^*}{\|x - x^*\|}, \tag{4.3}$$

and hence has a gradient with unitary Euclidean norm, i.e., $\|\nabla d_f^*(x)\| = 1$, referred to as *unitary gradient* (UG) for short in the following. Furthermore, d_f^* is such that:

1. It provides a trivial way to certify the robustness of $\hat{k}$ in x, since, by definition, $|d_f^*(x)|$ represents the MAP.
2. It explicitly provides the closest adversarial example to x, which can be computed $x^* = x - d_f^*(x)\nabla d_f^*(x)$.

Proof. Refer to [88, Thm. 4.8] □

Inspired by these intriguing properties, this work aims at *investigating classifiers whose output provides the distance (with sign) from their own classification boundary.*

4.2.2 A characterization property

A trivial example of a binary classifier f that coincides with a signed distance function is given by any *affine function* with a unitary weight. Indeed, if $f(x) = w^T x + b$, where $\|w\| = 1$, then the MAP relative to f has the explicit unique solution of the form $x^* = x - \frac{f(x)}{\|w\|^2} w$, as already pointed out in [6], from which $d_f(x) = |f(x)|$. As shown in Observation 1, a signed distance function has a unitary gradient. Under certain hypotheses, the opposite implication holds: a function f with a unitary gradient coincides with a signed distance function from $\mathcal{B}_f$. For the sake of clear comprehension, before providing the proof, let us remember some classical results. The following theorems are known as Existence and Uniqueness of Solutions of Ordinary Differential Equations (ODE) and Implicit Function Theorem.

Lemma 3 (Existence and Uniqueness of ODE solutions)**.** *Let $\mathcal{U} \subseteq \mathbb{R}^n$ be an open subset of $\mathbb{R}^n$, and let $F : \mathcal{U} \to \mathbb{R}^n$ a smooth function, i.e., $F \in C^\infty(\mathcal{U})$, then the following statements hold.*

i) For each $t_0 \in \mathbb{R}$ and $x_0 \in \mathcal{U}$, there exists $I_0 \subseteq \mathbb{R}$ and $\mathcal{U}_0 \subseteq \mathcal{U}$ open sets, with $(t_0, x_0) \in I_0 \times \mathcal{U}_0$, such that for each $x \in \mathcal{U}_0$ there exists a solution $u_x : I_0 \to \mathcal{U}$ of the following Cauchy-problem

$$\begin{cases} \dot{u}(t) = F(u(t)) \\ u(0) = x; \end{cases} \tag{4.4}$$

where we keep the notation u_x to highlight that x is the starting point of the solution of Problem 4.4.

ii) The map $\Theta : I_0 \times \mathcal{U}_0 \subseteq \mathcal{U}$, namely flux*, defined by $\Theta(t, x) := u_x(t)$, is in C^∞;*

iii) If u_x, v_x are two solutions of Equation (4.4), then $u_x \equiv v_x$;

Proof. Refer to [89, pp.66-88]. □

The second classical result useful during the proof of the Theorem is the Implicit Function Theorem.

Lemma 4 (Implicit Function Theorem (Dini))**.** *Let $G : \mathbb{R} \times \mathcal{U} \to \mathbb{R}$ be a smooth function defined on an open set $\mathbb{R} \times \mathcal{U}$. If $p \in \mathcal{U}$ is such that*

$$G(0, p) = 0 \quad \textit{and} \quad \frac{dG}{dt}(0, p) \neq 0,$$

then there exists an open set $\Omega \subseteq \mathcal{U}$ and a smooth function $\varphi : \Omega \to \mathbb{R}$ such that

$$\forall x \in \Omega, \quad G(\varphi(x), x) = 0. \tag{4.5}$$

Proof. The proof can be deduced by [89, Thm. 5.9], where $V := \mathbb{R}$, $U := \mathcal{U}$, $U_0 := \Omega$ and $(a, b) = (p, 0)$. □

Finally, we can leverage these results to prove the main theorem of the paper.

Theorem 2. *Let $\mathcal{U} \subseteq \mathbb{R}^n$ be an open set, and let $f : \mathbb{R}^n \to \mathbb{R}$ be a function, smooth in $\mathcal{U}$, such that $\mathcal{B}_f \subseteq \mathcal{U}$. If f has a unitary gradient in $\mathcal{U}$, then there exists an open set $\Omega_f \subseteq \mathcal{U}$ such that f coincides in Ω_f with the signed distance function from $\mathcal{B}_f$. Formally,*

$$\|\nabla f_{\restriction \mathcal{U}}\| \equiv 1 \quad \Rightarrow \quad \exists \Omega_f \subseteq \mathcal{U}, \quad f_{\restriction \Omega_f} \equiv d^*_{f \restriction \Omega_f}. \tag{4.6}$$

Proof. The proof is built upon [90, Prop.2.1]. Any trajectory $\gamma : [0,1] \to \mathcal{U}$ that solves the dynamical system $\dot{\gamma}(t) = \nabla f(\gamma(t)))$ coincides with the shortest path between the point $\gamma(0)$ and the hyper-surface $f^{-1}(\gamma(1))$. Note, that we want to prove that there exists an open set $\Omega_f \subseteq \mathcal{U}$ such that the unitary gradient property in $\mathcal{U}$ implies that $f(x) = d^*_f(x)$ for all $x \in \Omega_f$. The proof is divided into two main parts:

(i) Let us consider the following ordinary differential equation with initial condition (a.k.a. Cauchy problem)

$$\begin{cases} \dot{u}(t) = \nabla f(u(t)) \\ u(0) = x \end{cases} \tag{4.7}$$

where $x \in \mathcal{U}$. We show that there exists an open set $\Omega_f \supseteq \mathcal{B}_f$ such that each $x \in \Omega_f$ can be reached by a solution u_p of the Cauchy-problem (4.7), i.e., $\exists s \in \mathbb{R}$ such that $x = \Theta(s,p) := u_p(s)$ for some $p \in \mathcal{B}_f$;

(ii) We show that any trajectory of the flux corresponds the minimal geodetic (i.e., the shortest path) between the hyper-surfaces of the form $f^{-1}(t)$ and $\mathcal{B}_f$. This can be obtained by explicitly deducing a close form of f on Ω_f.

Let us start with the existence of such a Ω_f. Since f is smooth, then $F := \nabla f$ satisfies the hypothesis of Lemma 3, by which we can deduce that for each $p \in \mathcal{B}_f$ there exists an open set $I_p \times U_p \subseteq \mathbb{R} \times \mathcal{U}$ such that the flux

$$\begin{aligned} \Theta : I_p \times \mathcal{U}_p &\to \mathcal{U} \\ (t,x) &\mapsto u_x(t) \end{aligned} \tag{4.8}$$

is of class C^∞ (where remember that u_x is the solution of the ODE (4.4) with starting point in x). Let $G : I_p \times \mathcal{U}_p \to \mathbb{R}$ be the smooth function defined by $G(t,x) := f(\Theta(t,x))$. By Equation (4.7), $\frac{d\Theta}{dt}(0,p) = \dot{u}_p(0) = \nabla f(u_p(0))$ and $\Theta(0,p) = u_p(0) = p$, hence it is possible to observe that

$$G(0,p) = f(\Theta(0,p)) = f(p) = 0 \tag{4.9}$$

and that

$$\frac{dG}{dt}(0,p) = \nabla f(p)^T \frac{d\Theta}{dt}(0,p) = \nabla f(p)^T \nabla f(p). \tag{4.10}$$

We then deduce by the Implicit Function Theorem of Lemma 4 that there exists an open set $\Omega_p \subseteq \mathcal{U}_p$ such that

$$\forall x \in \Omega_p, \quad \exists t \in I_p \quad : \quad G(t,x) = 0, \tag{4.11}$$

from which

$$\forall x \in \Omega_p, \quad \exists t \in I_p \quad : \quad u_x(t) \in \mathcal{B}_f. \tag{4.12}$$

From the uniqueness of the solution stated in Lemma 3, this implies that, for each $x \in \Omega_p$, there exists $q \in \Omega_p \cap \mathcal{B}_f$ and an instant $t \in I_p$ such that $u_q(t) = x$. Finally, by considering $\Omega_f := \cup_{p \in \mathcal{B}_f} \Omega_p$, the first step of the proof is concluded. Now, we want to prove that the trajectory of the dynamic system coincides with the geodetic (the curve of minimal length) from any $x \in \Omega_f$ and for any B_p. Let $u_p : I_p \to \Omega_f$ be the solution of (4.4) with starting point in $p \in \mathcal{B}_f$, and let $x = u_p(s)$ be the point of the trajectory for $s \in I_p$. Let us consider a function $\gamma(t) := u_p(ts)$ of the form $[0, 1] \to \Omega_f$ to denote the curve that connects p and x. Observe that the length of γ can be found by considering the following formula

$$L(\gamma) := \int_0^1 \|\dot{\gamma}(t)\|\, dt = \int_0^1 |s| \|\dot{u}_p(t)\|\, dt. \tag{4.13}$$

Since $\|\dot{u}_p\| = \|\nabla f(u(t))\| = 1$ we can deduce that the length of γ is $L(\gamma) = |s|$. Let $\zeta : [0, 1] \to \mathcal{U}$ be any other curve that connects p and x. Observe that the following chain of inequalities holds

$$\begin{aligned} L(\zeta) = \int_0^1 \|\dot{\zeta}\|\, dt \geq \\ \geq \int_0^1 \left| \left\langle \dot{\zeta}(t), \nabla f(\zeta(t)) \right\rangle \right| dt \geq \\ \geq \left| \int_0^1 \frac{d}{dt} f(\zeta(t))\, dt \right| = |f(p) - f(x)| = |f(x)|, \end{aligned} \tag{4.14}$$

where the first inequality is a direct consequence of the Cauchy-Schwartz inequality ($\forall u, v \in \mathbb{R}^n$, $|\langle u, v\rangle| \leq \|u\|\|v\|$).

It remains to prove that $L(\zeta) \geq L(\gamma)$. To do so, let us observe that ff $p \in \mathcal{B}_f$ and $s \in I_p$, then $f(u_p(s)) = s$. Indeed, let $\varphi(s) = f(u_p(s))$ be the value of f on the trajectory of the flux. Since $\dot{\varphi} = \langle \dot{u}_p(s), \nabla f(u_p(s))\rangle = 1$ we deduce $\varphi(s) = s + \varphi(0) = s$. This concludes the second step of the proof, since for each curve ζ that connects p and x we have that

$$L(\zeta) \geq |f(x)| = |s| = L(\gamma),$$

hence γ is the shortest path between p and x, from which $|f(x)| = d_f(x)$. In conclusion, the theorem is proved by observing that, for each $x \in \Omega_f$, there exists $p \in \mathcal{B}_f$ such that $x = u_p(s)$ for some s. Indeed, by the definition of Ω_f, let q such that $x \in \Omega_q$, then there exists a $p \in \Omega_q \cap \mathcal{B}_f$ such that $x = u_p(s)$ for some s. □

It is worth noting that, as pointed out in [90, Prop.2.1], the same characterization holds for particular geometrical spaces, i.e., *Complete Riemannian Manifolds*. Unfortunately, as shown by the author, the only smooth functions with a unitary gradient in a Complete Riemannian Manifold with non-negative Ricci Curvature (e.g., $\mathbb{R}^n$) are the affine functions [90, Theorem A]. However, an open set $\mathcal{U} \subset \mathbb{R}^n$ is a Riemannian Manifold that does not satisfy the completeness property. Hence, the existence of a non-affine signed distance function is not in contradiction with [90, Theorem A]. A trivial example is given by the binary classifier $f(x) = \|x\| - 1$ defined in $\mathcal{U} = \mathbb{R}^n \setminus \{0\}$. Further details are provided in the Appendix.

4.2.3 Extension to multiclass classifiers

By following the *one-to-rest* strategy [91], the results above can be extended to multi-class classifiers. Let $f : \mathbb{R}^n \to \mathbb{R}^C$ be a smooth function by which the predicted class of a sample $x \in \mathbb{R}^n$ is given by $\hat{k}(x) = \operatorname{argmax}_i f_i(x)$, where $\hat{k}(x) = 0$ if there is no unique maximum component. Remember that, as discussed in the previous chapter, and according to [1], [2], [6], the MAP problem for a multi-class classifier can be stated as follows:

$$\begin{aligned} d_f(x) := \inf_{p \in \mathbb{R}^n} & \quad \|p - x\| \\ \text{s.t.} & \quad \hat{k}(p) \neq \hat{k}(x). \end{aligned} \tag{MAP}$$

Here, we extend the definition of signed distance function d_f^* to a multi-class *Signed Distance Classifier* f as follows.

Definition 4 (Signed distance classifier)**.** A function $f : \mathbb{R}^n \to \mathbb{R}^C$ is a *Signed Distance Classifier* if, for each pair i, j, with $i \neq j$, the difference $(f_i - f_j)$ corresponds to the signed distance from the one-to-one classification boundary $\mathcal{B}_{ij} := \{x \in \mathbb{R}^n \, : \, f_i - f_j = 0\}$.

The following observation shows that an SDC satisfies similar properties of Observation 1 for binary classifiers. In particular, the characterization property can be extended to the multiclass case. Before providing the formal statement, let us introduce the following Lemma.

Lemma 5. *Let $x \in \mathbb{R}^n$ classified from f with the class l, $\hat{k}(x) = l$. Let, for each $j \neq l$, $g_j = (f_l - f_j)$, then*

$$d_f(x) = \min_{j \neq l} d_{g_j}$$

where d_{g_j} is the solution of the Problem 4.1 relative to the binary classifier g_j. In formulas,

$$\begin{aligned} d_{g_j}(x) := \inf_{p \in \mathbb{R}^n} & \quad \|p - x\| \\ s.t. & \quad f_l(p) - f_j(p) = 0 \end{aligned} \tag{4.15}$$

Proof. The main idea is to prove the two inequalities

$$\min_{j \neq l} d_{g_j}(x) \leq d_f(x) \leq \min_{j \neq l} d_{g_j}(x). \tag{4.16}$$

The inequality on the right can be deduced by observing that, for each j, the solution x_j^* of the Problem 4.1, relative to the function g_j, satisfies the constraints of the minimum problem MAP relative to the function f. Hence, by the definition of minimum $d_f(x) = \|x - x^*\| \leq \|x - x_j^*\|$. The inequality on the left is deduced by observing that if x^* is the solution of d_f and if j^* is such that $f_{j^*} = \max_{j \neq l} f_j(x^*)$, then x^* satisfies the constraints of the Problem 4.1 for the function $d_{g_{j^*}}$. Hence,

$$\min_{j \neq l} d_{g_j} \leq d_{g_{j^*}} \leq \|x - x^*\|,$$

which concludes the proof. □

Observation 2. Let $f : \mathbb{R}^n \to \mathbb{R}^C$ be a signed distance function and let $x \in \mathbb{R}^n$ be a sample classified as $l = \hat{k}(x)$. Let $s := \operatorname{argmax}_{j \neq l} f_j(x)$ be the second-highest component of $f(x)$. Hence, the classifier f:

1. Provides a fast way to certificate the robustness of $\hat{k}$ in x. In fact, $f_l(x) - f_s(x) = d_f(x)$, where $d_f(x)$ is the MAP.

2. Provides the closest adversarial example to x, i.e.,

$$x^* = x - (f_l(x) - f_s(x))\nabla(f_l - f_s)(x),$$

where x^* is the unique solution of Problem MAP in x.

Proof. The first statement is a direct consequence of Lemma 5. Consider the following chain of equalities

$$\begin{aligned} d_f(x) &= \min_{j \neq l} d_{g_j}(x) = \min_{j \neq l}(f_l - f_j)(x) \\ &= f_l - \max_{j \neq l} f_j(x) = (f_l - f_s)(x). \end{aligned} \tag{4.17}$$

where the second equivalence is given by the definition of a signed distance classifier. The second statement is a consequence of Observation 1, indeed, $\nabla(f_l - f_s)(x)$ provides the direction of the shortest path to reach B_{ls}. □

Similarly to the binary case, an SDC is characterized by having a *unitary gradient* for each pair-wise difference of the output components. In details, by directly applying Theorem 2, a smooth classifier f is a signed distance classifier (in some open set) if and only if $\|\nabla(f_i - f_j)\| \equiv 1, \forall i \neq j$.

4.3 Unitary-gradient neural networks

In the previous section, we showed that if a smooth classifier f satisfies the unitary gradient property in some open set $\mathcal{U} \supseteq \mathcal{B}_f$, then it admits an open set $\Omega_f \supseteq \mathcal{B}_f$ in which f coincides with the signed distance function with respect to the boundary $\mathcal{B}_f$. Furthermore, affine functions represent all and the only smooth SDCs in the whole $\mathbb{R}^n$. Supported by these results, any DNN that globally satisfies the UG property would coincide with a trivial linear model, which hardly provides good classification performance for complex tasks. To approximate a non-trivial SDC with a well-performing DNN f_θ, we impose the UG property almost everywhere. This section shows the proper requirements on f_θ to satisfy the hypothesis of Theorem 2, providing layer-wise sufficient conditions that ensure the UG property. To this end, we focus our analysis on the family $\mathcal{F}$ of feed-forward DNNs $f : \mathbb{R}^n \to \mathbb{R}^C$ of the form $f = g \circ h^{(L)} \circ \cdots \circ h^{(1)}$, where g is the output-layer and each $h^{(i)}$ is any canonical elementary layer (e.g., Fully Connected, Convolutional, etc.) or an activation function.

Observation 3 (Layer-wise sufficient conditions)**.** Let f be a DNN in $\mathcal{F}$. For each i, let $J^{(i)}(x)$ be the Jacobian of $h^{(i)}$ evaluated in $y = h^{(i-1)} \circ \cdots \circ h^{(1)}(x)$. For each $j = 1, \ldots, C$, let $W_j(x)$ be the Jacobian of g_j evaluated in $y = h^{(L)} \circ \cdots \circ h^{(1)}(x)$. Hence, if

$$J^{(i)} J^{(i)T} \equiv I, \quad \forall i = 1, \ldots, L \tag{GNP}$$

$$(W_h - W_k)(W_h - W_k)^T \equiv 1, \quad \forall h \neq k, \tag{UPD}$$

then, for all $h \neq k$, $f_h - f_k$ satisfies the UG property.

Proof. For a feed-forward neural network, the Jacobian matrix of each component f_j can be decomposed as

$$\mathrm{Jac}(f_j) = W_j \prod_{i=1}^{L} J^{(i)} = W_j J^{(L)} \cdots J^{(1)}. \tag{4.18}$$

Hence, the thesis follows by the associative property and by observing that $(AB)^T = B^T A^T$ for any two matrices. □

Observe that Condition GNP, namely *Gradient Norm Preserving*, requires any layer to have an output dimension no higher than the input dimension. Indeed, a rectangular matrix $J \in \mathbb{R}^{M \times N}$ can be orthogonal by row, i.e., $JJ^T = I$, only if $M \leq N$. Condition GNP is also addressed in [55], [56] to build Lipschitz-Bounded Neural Networks. However, for their purposes, the authors also consider DNNs that satisfy a weaker condition named *Contraction Property* (see [56]), which includes the $M \geq N$ case.

4.3.1 Gradient norm preserving layers

We now provide an overview of the most common layers that can satisfy the GNP property. For a shorter notation, let h be a generic internal layer.

Activation functions Activation functions $h : \mathbb{R}^n \to \mathbb{R}^n$ can be grouped in two main categories: *component-wise* and *tensor-wise* activation functions. Common component-wise activation functions as ReLU, tanh, and sigmoid do not satisfy the GNP property [82]. Moreover, since any component-wise function h that satisfies the GNP property is piece-wise linear with slopes ± 1, the absolute value function and the recent $\mathcal{N}$-activation function, [92], are examples of GNP component-wise activations. Tensor-wise activation functions have recently gained popularity thanks to [58], [79], [82], who introduced OPLU, GroupSort, HouseHolder activation functions, respectively, which are specifically designed to satisfy the GNP property. An overview of these activation functions is left in the appendix. In this work, we compare the abs with the OPLU and the GroupSort with a group size of 2, a.k.a MaxMin.

Dense and Convolutional layers A fully connected layer of the form $h(x) = Wx+b$ has a constant Jacobian matrix $\mathrm{Jac}(h)(x) = W$. This implies that h is GNP if and only if W is an orthogonal-by-row matrix. Similarly, for a convolutional layer with kernel $\mathcal{K}$ of shape $M \times C \times k \times k$, the GNP property can be satisfied only if $M \leq C$, i.e., the layer does not increase the number of channels of the input tensor [86]. As done in [58], in our model, we consider the Björck parameterization strategy to guarantee the orthogonality of the fully connected layers and the CayleyConv strategy presented in [56] for the convolutional layers.

Pooling, normalization and residual layers Tow dimensional maxpooling layers with kernel $\boldsymbol{k} = (k_1, k_2) \in \mathbb{N}^2$, stride $\boldsymbol{s} = \boldsymbol{k}$, and without padding, satisfy the GNP property if applied to a tensor whose spatial dimensions H, W are multiples of k_1 and k_2, respectively. This can be proved by observing that the Jacobian matrix corresponds to an othogonal projection matrix [86]. Batch-normalization layers with a non-unitary variance do not satisfies the GNP property [86]. For residual blocks, it is still not clear whether they can or cannot satisfy the GNP property. Indeed, a residual layer of the form $h(x) = x + \tilde{h}(x)$ is GNP if and only if $\text{Jac}(\tilde{h})\text{Jac}(\tilde{h})^T + \text{Jac}(\tilde{h}) + \text{Jac}(\tilde{h})^T \equiv 0$. For such reasons, the last two mentioned layers are not considered in our model.

4.3.2 Unitary pair difference layers

This section focuses on the second condition stated in Observation 3: the *Unitary Pair Difference* (UPD). Since most neural classifiers include a last dense layer, we restrict our analysis to this case. Let $g(x) = Wx + b$ be the last layer since $\text{Jac}(g) \equiv W$, then the UPD property requires that for every two rows W_h, W_k the difference $W_h - W_k$ has unitary norm. A matrix W satisfies the UPD property if the function $x \mapsto Wx$ is UPD.

Bounded UPD layer. An UPD matrix from any orthogonal-by-row matrix as stated by the next observation.

Observation 4. Let $Q \in \mathbb{R}^{m\times C}$ such that $QQ^T = I$. Then, $W = \frac{1}{\sqrt{2}}Q$ satisfies the UPD property, indeed

$$\|W_h - W_k\|^2 = \underbrace{\|W_h\|^2}_{1/2} + \underbrace{\|W_k\|^2}_{1/2} - 2\underbrace{W_h^T W_k}_{0} = 1. \tag{4.19}$$

An UPD layer with matrix W as above is said to be *bounded*, as each row of W is bounded to have norm $1/\sqrt{2}$.

As pointed out in [79], this constraint makes it harder to train the model when the output dimension C is large (i.e., there are many classes).

Unbounded UPD layer. To avoid this issue, we considered an UPD layer with parametric weight matrix $W(U)$. Matrix $W(U)$ is obtained by iteratively applying the L-BFGS, an optimization algorithm for unconstrained minimum problems [93], to the loss

$$\Psi(U) = \sum_{h<k}(\|U_h - U_k\|^2 - 1)^2. \tag{4.20}$$

More specifically, if `psi` is the routine that computes such a loss function and `L-BFGS` is the routine that performs one step of the L-BFGS optimization algorithm, then the weight matrix is obtained as W = `UPD`(U), where `UPD` is the following procedure:

```
def UPD(U: Tensor):
    # Returns an UPD matrix
    W = U
    for in range(max_iter):
        W = L-BFGS(psi(U), W)
    return W
```

Listing 4.1: Psudo code that parameterizes an UPD matrix through a parameter U. The resulting W is obtained by performing few steps of the L-BFGS algorithm to find a minimum of Ψ with starting point U.

Note that the UPD layer $g(x) = W(U)\,x+b$ depends on the weights U, b and it is fully differentiable in U. This implies that such a procedure, like parameterization strategies for orthogonal layers, can be applied during training. Finally, note that the algorithm complexity strongly depends on the computational cost of the objective $\Psi(U)$. Our implementation exploits parallelism by implementing $\Psi(U)$ by means of a matrix product of the form $A^{(C)}U$, where $A^{(C)}$ is designed to compute all the pair differences between rows required by Eq. (4.20). In detail, the objective function Ψ can be efficiently computed by exploiting the parallelism as follows. Let us consider the family of matrices $A^{(k)}$, within $\binom{k}{2}$ rows and k columns, recursively defined as follows

$$A^{(2)} = \begin{pmatrix} 1 & -1 \end{pmatrix}, \quad A^{(k)} = \left(\begin{array}{c|c} 1 & \\ \vdots & -Id_{k-1} \\ 1 & \\ \hline 0 & \\ \vdots & A^{(k-1)} \\ 0 & \end{array}\right) \quad \forall k \geq 2. \tag{4.21}$$

Hence, observe that if $U \in \mathbb{R}^{C\times m}$ is some matrix, then the resulting matrix product $A^{(C)}\,U$ corresponds to a matrix where each row is one of the possible difference between two rows of U. In formulas

$$A^{(C)}U = A^{(C)} \begin{pmatrix} U_1^T \\ \vdots \\ U_C^T \end{pmatrix} = \begin{pmatrix} (U_1 - U_2)^T \\ \vdots \\ (U_1 - U_C)^T \\ (U_2 - U_3)^T \\ \vdots \end{pmatrix}. \tag{4.22}$$

This allows exploiting the parallelism of the GPUs in order to efficiently compute the objective function Ψ. In conclusion, experimental tests reported in Figure 4.2 show that 3 iterations of the L-BFGS algorithm are sufficient to obtain a UPD matrix W whose differences between rows have an Euclidean norm in the range 1 ± 10^{-5} for the case of interest ($C = 10$).

4.3.3 Unitary-Gradient Neural Network Architecture

This section describes how to practically combine GNP and UPD layers to obtain a neural network f_θ such that all pair-wise differences of its output vector have unitary gradient. The main difficulty in crafting such a network is due to the GNP property, which implies a decreasing dimension in both dense and convolutional layers. Indeed, most DNNs for image classification process a 3-channel image by gradually increasing the channel dimension of convolutional layers. To overcome this issue, we leverage a 2-dimensional *PixelUnshuffle* layer [94], which inputs a tensor of shape $C \times rH \times rW$ and outputs a tensor of shape $r^2C \times H \times W$. The output is obtained by only rearranging input components. As such, this layer satisfies the GNP property. The PixelUnshuffle layer allows increasing the

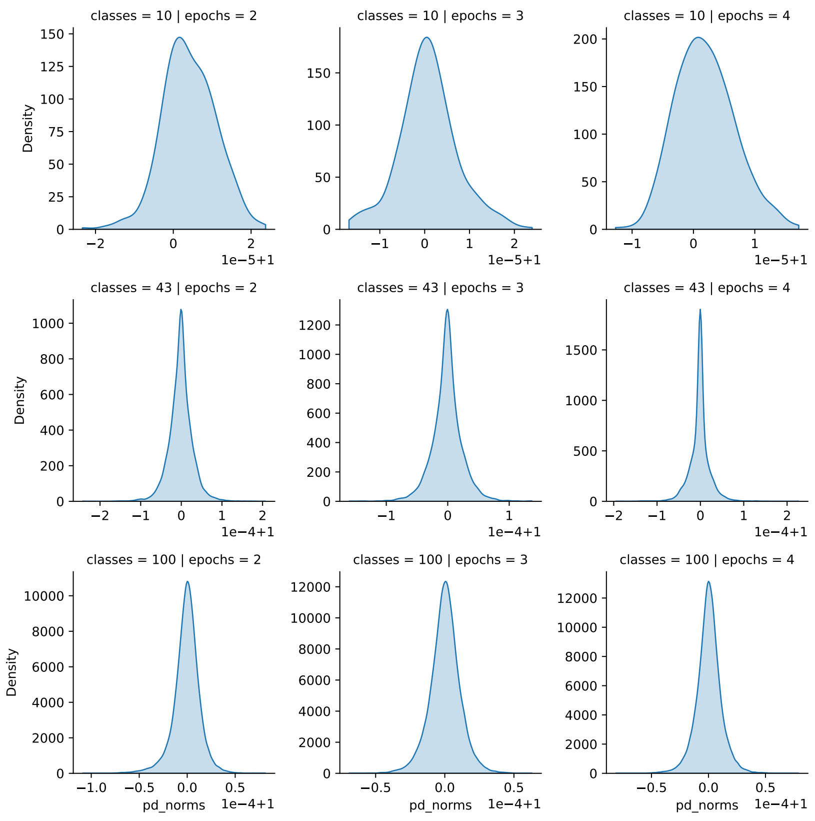


Figure 4.2: Distribution of the pair row differences of weight matrices $W = \texttt{UPD}(U)$ obtained by applying the L-BFGS for $2, 3, 4$ steps. The analysis involves matrices with $10, 43, 100$ rows and 512 columns. Distributions are computed by evaluating the Euclidean norm of all the pair-wise differences of the rows of the matrix W for 10 random generated parameters U.

number of channels of hidden layers even in convolutional GNP networks. It is worth pointing out that [55], [56] also leveraged such a permutation layer, but only to emulate a convolution with stride 2. That said, the UGNN proposed in this work, shown in Fig. 4.3, consists of five GNP blocks, two fully connected GNP layers, a last UPD layer (bounded or unbounded), and GNP activation functions. Each GNP block consists of two GNP convolutional layers and one last PixelUnshuffle layer with scaling factor 2; a GNP activation function is applied after each convolution (see Tab. 4.1). Each convolutional layer has a circular padding to preserve the spatial dimension of the input. Furthermore, before the flattening stage, a max-pool layer with window size and stride $H/2^5$ is applied to process input of different spatial dimensions $H = m \cdot 2^5$, for any $m \in \mathbb{N}$. Note that, to the best of our records, this is the first instance of a convolutional DNN that aims at practically implementing an SDC and that provably satisfies $\|\nabla(f_i - f_j)\| \equiv 1$ almost everywhere. [95] only focused on fully-connected networks, while [81] only approximated an optimal f^* such that $\|\nabla f_i^*\| \equiv 1$. [96] approximate a binary SDC via SVM. In conclusion, observe that, by design, each pair-difference $f_i - f_j$ of an UGNN satisfies the 1-Lipschitz property, hence the margin $\mathcal{M}_f(x) = f_l(x) - \max_{j \neq l} f_j(x)$ is a lower bound of the MAP in x.

Observation 5 (Certifiable Robustness)**.** If f is a UGNN, then $\hat{k}(x) = \mathrm{argmax}_i\, f_i(x)$

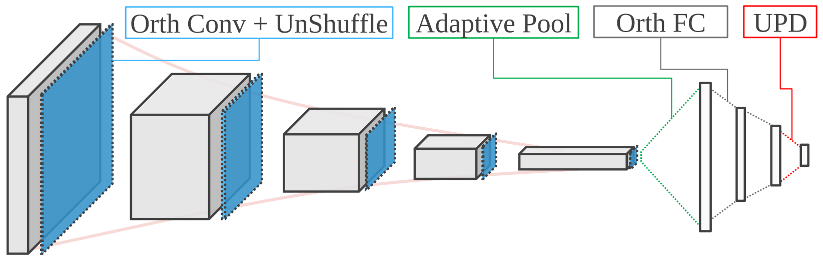


Figure 4.3: Tested UGNN architecture: 5 GNP conv-blocks, 2 FC GNP layers and 1 UPD layer.

is $\mathcal{M}_f(x)-$robust in x.

Proof. The proof can be easly deduced by observing that, by construction, each pair-difference $f_i - f_j$ of an UGNN satisfies the 1-Lipschitz property, hence the margin $\mathcal{M}_f(x) = f_l(x) - \max_{j \neq l} f_j(x)$ is a lower bound of the MAP in x, [56]. □

4.4 Experimental Results

Experiments were conducted to evaluate the classification accuracy of a UGNN and its capability of implementing an SDC. As done by related works, the experiments targeted the CIFAR10 datasets. We compared UGNN with the following 1-Lipschitz models: *LargeConvNet* [55], *ResNet9* [56], and *LipConvNet5* [59].

4.4.1 Experimental Setup

For all the combinations of GNP activations, UPD layers, preprocessing, and input size, our model was trained for 300 epochs, using the Adam optimizer [38], with learning rate decreased by 0.5 after 100 and 200 epochs, and a batch of 1024 samples, containing randomly cropped and randomly horizontally flipped images. This configuration is the same used in [59]. The other models were trained by following the original papers, leveraging a multi-margin loss function with a margin $m = \varepsilon\sqrt{2}$, with $\varepsilon = 0.5$. For a fair comparison, UGNN was trained with a margin $m = \varepsilon$, being the lower bound $\mathcal{M}_f$ of the MAP for UGNN different from the $\mathcal{M}_f/\sqrt{2}$ of the other DNNs, as discussed in Observation 5. For the experiments, we used 4 Nvidia Tesla-V100 with cuda 10.1 and PyTorch 1.8 [73].

Layers	**Output Shape**
`OrthConv`$(3\cdot 4^i, 3\cdot 4^i, 3)$	$3\cdot 4^i \times \frac{H}{2^i} \times \frac{H}{2^i}$
`GNP Activation`	-
`OrthConv`$(3\cdot 4^i, 3\cdot 4^i, 3)$	-
`GNP Activation`	-
`PixelUnshuffle`(2)	$3\cdot 4^{i+1} \times \frac{H}{2^{i+1}} \times \frac{H}{2^{i+1}}$

Table 4.1: An example of the (i+1)th internal GNP conv-block. Observe that the number of channels increases with a geometric progression of common ratio 4 and each spatial dimension decreases with a ratio of 2.

4.4.2 Accuracy Analysis

Table 4.2 summarizes the accuracy on the testset, where UGNN was tested with the (Abs, MaxMin, OPLU) activation, and the last UPD layers (bounded and unbounded). The other models were tested with the original configuration and with

	Accuracy [%]	
Models	**Std.Norm**	**Raw**
LargeConvNet	**79.0** $\pm$ 0.26	**72.2** $\pm$ 0.11
LargeConvNet+Abs	77.8 $\pm$ 0.33	71.8 $\pm$ 0.25
LipConvNet5	78.0 $\pm$ 0.26	68.8 $\pm$ 0.35
LipConvNet5+Abs	76.1 $\pm$ 0.31	65.5 $\pm$ 0.68
ResNet9	78.7 $\pm$ 0.22	66.4 $\pm$ 0.17
ResNet9+Abs	78.1 $\pm$ 0.34	65.6 $\pm$ 0.22
UGNN+Abs+updB	71.9 $\pm$ 0.29	69.2 $\pm$ 0.31
UGNN+Abs+updU	72.1 $\pm$ 0.54	68.9 $\pm$ 0.81
UGNN+MaxMin+updB	72.6 $\pm$ 0.79	70.4 $\pm$ 0.52
UGNN+MaxMin+updU	**72.7** $\pm$ 0.38	70.4 $\pm$ 0.86
UGNN+OPLU+updB	71.9 $\pm$ 0.09	70.5 $\pm$ 0.39
UGNN+OPLU+updU	72.0 $\pm$ 0.70	**70.6** $\pm$ 0.45

Table 4.2: Accuracy comparison between the 1-Lipschitz models and the UGNNs.

the abs activation. Experiments were performed with and without standard normalization (Std.Norm) of the input, and each configuration was trained four times with randomly initialized weights to obtain statically sound results. In summary, the take-away messages of the Tab. 4.2 are: (i) The unbounded UPD layer (named updU) increased the performance with respect to the bounded one (named updB) in almost all cases. (ii) Std.Norm pre-processing significantly increased the performance. We believe this is due to the GNP property of the layers, which cannot learn a channel re-scaling different from ± 1. (iii) The use of abs activations in the 1-Lipschitz models does not cause a significant performance loss with respect to the other GNP activations (that requires a more expensive sorting). (iv) Despite the strict constraints of the UGNN architecture, it achieves comparable performance in the raw case, while there is a clear gap of accuracy for the Std.Norm case.To improve the UGNN accuracy, we investigated for intrinsic learning characteristics of its architecture. In particular, we noted that a strong limitation of the model is in the last two GNP blocks (see Fig. 4.3), which process tensors with a high number of channels (thus higher learning capabilities) but with compressed spatial dimensions ($H/8$ and $H/16$). Hence, for small input images (e.g., 32×32), such layers cannot exploit the spatial capability of convolutions. Table 4.3 reports a performance evaluation of the UGNN (with MaxMin activation) for larger input sizes. Note that, differently from the UGNN, common DNNs do not benefit from an up-scaling image transformation, since it is possible to apply any number of channels on the first convolutional layers. Moreover, the compared models do not have adaptive layers, hence do not handle different input sizes. This observation

		Accuracy [%]	
Input Size	**Last Layer**	**Std.Norm**	**Raw**
64	updB	72.1 ± 0.27	72.4 ± 0.42
	updU	72.6 ± 0.69	72.8 ± 0.61
128	updB	74.5 ± 0.56	75.9 ± 0.07
	updU	74.9 ± 0.45	76.2 ± 0.30
256	updB	76.5 ± 0.35	78.4 ± 0.29
	updU	$\mathbf{76.8} \pm 0.29$	$\mathbf{78.5} \pm 0.22$

Table 4.3: Accuracy comparison of the UGNN models with different, pre-processed input sizes and output layers.

allows the UGNN to outperform the other models for the raw case and reach similar accuracy for the Std.Norm case.

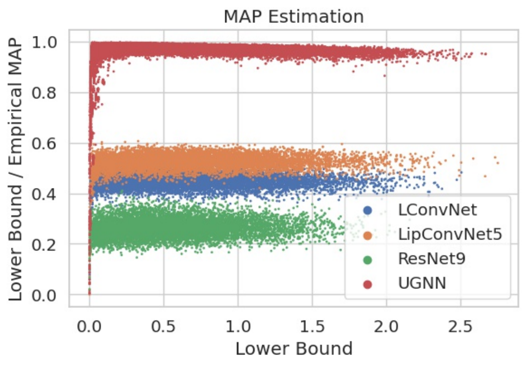
4.4.3 MAP estimation through UGNN

This section evaluates the MAP estimation through the lower bound (LB) given by the UGNN discussed in Observation 2 and the other 1-Lipschitz models.

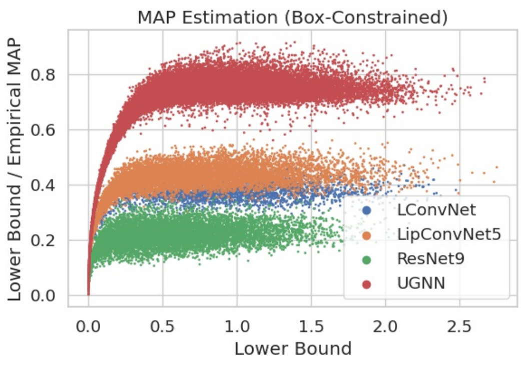
Figure 4.4 compares the ratio of the LB and the MAP between the 1-Lipschitz DDNs and a UGNN with MaxMin and bounded upd in four different scenarios. The MAP is computed with the expensive Iterative Penalty procedure described in Section 2.4. Note that our analysis considers the worst-case MAP, i.e., without *box-constraints*, as also done by the compared 1-Lipschitz models. Indeed, since image pixels are bounded in $[0, 1]$, the MAP is itself a lower bound of the distance from the closest adversarial image. Table 4.4 reports statistics related to the LB/MAP ratio for different UGNNs, where the box-constrained (BC) MAPs were computed using the Decoupling Direction Norm strategy [9]. The column `#N` contains the number of samples correctly classified by the model and for which the MAP algorithm reached convergence. Note that, in all the tested cases, the LB provided by the UGNN resulted to be tighter than the other 1-Lipschitz DNNs. Similar considerations hold for other model configurations (see Appendix).

4.4.4 Certifiable robust accuracy

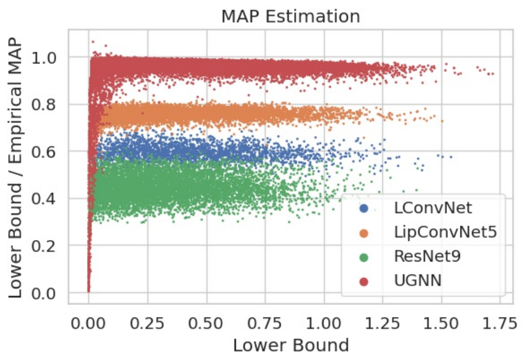
Figure 4.5 shows a close comparison of the certifiable robust accuracy of the for different values of ε, i.e., the percentage of correctly classified samples with a LB lower than ε (refer to Chapter 3 for a detailed definition). Unfortunately, even though UGNN clearly provides a better approximation of the MAP, the certifiable accuracy of the model does not outperform the certifiable accuracy of other 1-Lipschitz models. Indeed, as it can be observed from Figure 4.5, the same relative drop in the accuracy of the other 1-Lipschitz models can be noticed for the UGNN, suggesting also that the certifiable robustness is strictly related to its (low) standard accuracy.



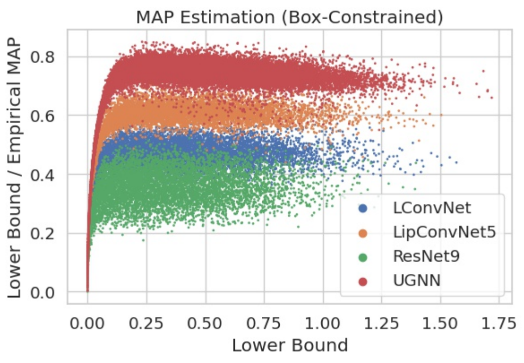
(a) Normalized Inputs. UGNN with OPLU and upd.



(b) Normalized Inputs. UGNN with OPLU and upd.



(c) Raw Inputs (not normalized). UGNN with MaxMin and orth.



(d) Raw Inputs (not normalized). UGNN with MaxMin and orth.

Figure 4.4: Comparisons of MAP estimation for the tested models with and without box constraints. For the empirical MAP computation, we used IP (a, c) for the Box-Unconstrained case and DDN for the constrained (b,d). Note that, in the image (c), due to a failure in the IP algorithm convergence, a few samples (less than 20) reported an inconsistent MAP lower than expected.

Model	**LB/MAP**	**#N**	**B.C**
ResNet9 (norm)	.21±.042	7900	✓
ResNet9 (raw)	.34±.063	6669	✓
LargeConvNet (norm)	.36±.036	2148	✓
LipConvNet5 (norm)	.41±.060	7838	✓
LargeConvNet (raw)	.46±.057	7219	✓
LipConvNet5 (raw)	.58±.069	6911	✓
UGNN+OPLU+updB (norm)	**.67±.129**	7101	✓
UGNN+Abs+updB (norm)	**.67±.129**	7220	✓
UGNN+OPLU+updU (norm)	**.67±.131**	7281	✓
UGNN+Abs+updU (norm)	**.67±.128**	7244	✓
UGNN+MaxMin+updU (norm)	**.67±.130**	7311	✓
UGNN+MaxMin+updB (norm)	**.69±.109**	4386	✓
UGNN+OPLU+updU (raw)	**.70±.090**	7125	✓
UGNN+OPLU+updB (raw)	**.70±.093**	7098	✓
UGNN+MaxMin+updB (raw)	**.71±.087**	7114	✓
UGNN+MaxMin+updU (raw)	**.71±.088**	7118	✓
UGNN+Abs+updB (raw)	**.71±.090**	6960	✓
UGNN+Abs+updU (raw)	**.71±.092**	6940	✓
ResNet9 (norm)	.26±.036	7904	✗
ResNet9 (raw)	.44±.056	6663	✗
LargeConvNet (norm)	.44±.027	7933	✗
LipConvNet5 (norm)	.52±.031	7840	✗
LargeConvNet (raw)	.58±.046	2429	✗
LipConvNet5 (raw)	.74±.049	6912	✗
UGNN+OPLU+updU (raw)	**.93±.101**	6755	✗
UGNN+OPLU+updB (raw)	**.95±.063**	7102	✗
UGNN+MaxMin+updU (raw)	**.95±.061**	7127	✗
UGNN+MaxMin+updB (raw)	**.95±.056**	7117	✗
UGNN+Abs+updB (raw)	**.95±.058**	6965	✗
UGNN+Abs+updU (raw)	**.95±.058**	6949	✗
UGNN+OPLU+updB (norm)	**.96±.046**	7215	✗
UGNN+Abs+updB (norm)	**.96±.049**	7228	✗
UGNN+OPLU+updU (norm)	**.96±.047**	7282	✗
UGNN+MaxMin+updU (norm)	**.96±.051**	7316	✗
UGNN+MaxMin+updB (norm)	**.96±.044**	7327	✗
UGNN+Abs+updU (norm)	**.96±.039**	7247	✗

Table 4.4: Evaluation of the LB/MAP ratio deduced by the output of the models with/without Box Constraint.

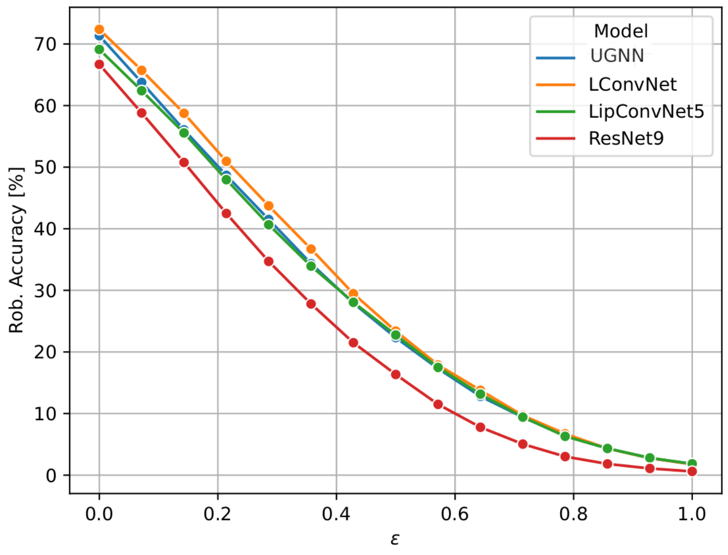


Figure 4.5: Certifiable robust accuracy for different values of ε despite the highest precision of the UGNN model in approximating the actual minimal adversarial perturbation, the certifiable robust accuracy does not outperform other 1-Lipschitz models.

Chapter 5

Conclusions and Future Works

This thesis delved into various methods for obtaining formal guarantees on the robustness of deep neural networks. This chapter aims to summarize the main contributions of each single chapter of the thesis and provide some insights into future research directions.

5.1 Summary of the contributions

The first strategy for achieving certifiable robustness for deep neural networks is based on the fast estimation of the minimal adversarial perturbation presented in the Chapter 2. In this chapter, the problem of estimating the minimal adversarial perturbation is addressed, proposing two root-finding-based strategies and providing theoretical results on the goodness of the estimation. Such a theoretical finding can be leveraged to estimate the robustness of a classifier for a given input x close enough to the classification boundary. Indeed, the approximate distance $t(x,l)$, obtained with the proposed approaches, results in being less computationally expensive than the distance $d(x,l)$ obtained with the state-of-the-arte methods, enabling a fast estimation of the ε-robustness of a classifier for the sample x. Furthermore, the goodness of the estimation is linked to a model-dependent value, named boundary proximity index $\mathcal{K}_f(\Omega_\delta)$, which encapsulates the regularity of the decision boundary. Indeed, such a coefficient depends on the first and the second derivative of the model and provides a neighborhood from which the decision boundary can be easily reached only by following the gradient direction; the larger the BPI the larger the neighborhood.

Observe that a sufficient condition for a large BPI of a certain model f is that the gradient ∇f has a constant Euclidean norm, and the eigenvalues of the Hessian matrix $\nabla^2 f$ are bounded. Despite it is not completely clear how to design a model with a large BPI, signed distance classifiers, presented in the Chapter 4, represent a natural attempt since feature a constant gradient of unitary Euclidean norm. Nevertheless, signed distance classifiers require orthogonal convolutional and dense layers, and hence are strictly related to the 1-Lipschitz neural networks that can be obtained by composing accurately Lipschitz-bounded layers.

For this reason, Chapter 3 focused on addressing certifiable guarantees of robustness for deep neural networks through Lipschitz bounded layers, by presenting a comparative study of state-of-the-art 1-Lipschitz layers. The comparison is made under the lens of different metrics, such as time and memory requirements, accuracy, and certified robust accuracy, all evaluated at training and inference time. A theoretical comparison of the methods in terms of time and memory complexity was also presented and validated by experiments. Taking all metrics into account (summarized in Figure 3.1), the results are in favor of CPL, due to its highest performance and lower consumption of computational resources. When large computational resources are available and the application does not impose stringent timing constraints during inference and training, the SOC layer could be used, due to its slightly better performance. Finally, those applications in which the inference time is crucial may take advantage of AOL or BCOP, which do not introduce additional runtime overhead (during inference) compared to a standard convolution.

Finally, the Chapter 4, deepen another possible direction for enhancing certifiable robst guarantees. This chapter proposed the novel family of *Signed-Distance Classifiers* (SDCs), which provides the *minimal adversarial perturbation* (MAP) by just computing the difference between the two highest output components, thus offering an online-certifiable prediction. To practically approximate an SDC, we considered a neural network model, named *Unitary-Gradient Neural Network* (UGNN), that furthermore involves a novel dense layer, named *Unitary Pair Difference (UPD)*, which features an unbounded weight matrix while preserving the unitary-gradient property. Several experiments were conducted to compare the proposed architecture with the most related certifiable 1-Lipschitz models. On one hand, the experiments highlighted that the UGNN is capable of providing a strict estimation of the MAP, outperforming the lower bound provided by the other models. On the other hand, the UGNN showed a low accuracy in the classification task, in line with the other 1-Lipschitz models composed of orthogonal convolutional and dense layers. We believe this is a consequence of the strict constraints of the architecture that limit the learning capabilities of the model.

5.2 Impact of the contributions

The fast estimation of the minimal adversarial perturbation, presented in Chapter 2, practically was conceived as a fast verification-like method that could be deployed on a safety-critical system with a low computational budget. An open-source repository containing available code for experiments can be found in gitlab.retis.santannapisa.it/retis-ai/fast-map. The analysis of the curvature of the decision boundary proposed in the paper [14] has been appreciated in the work of [97], which proposes a revisited version of the deepfool method. The revised method takes into account the direction of the gradient with respect to the decision boundary while finding the adversarial direction, thus providing a more accurate estimation of the minimal adversarial perturbation.

The *signed distance classifers* proposed in the Chapter 4 practically was conceived to obtain a model (the *unitary gradient neural network*) that, similarly to 1-Lipschitz models, directly outputs a lower bound of the minimal adversarial perturbation. Differently from Lipschitz bounded neural networks, the estimation of the MAP is tight, reducing the amount of ”false-postive” rejections. The results of the chapter are taken from the paper, [18], that has been considered in the works [98], [99] where the authors focused on *1-class classification* for the sake of out-of-distribution detection through SDC.

5.3 Takeways and future directions

The section summarizes the key takeaways of the thesis and proposes some future research directions that can be addressed in future works. In detail, the key takeaway of the Chapter 2 is the following. The fast estimation of the MAP for the sake of a robust classifier through rejection of unsafe inputs, without any other countermeasure, may be not a viable strategy. Indeed, the certifiable robust accuracy of such a classifier would be particularly low, since deep neural networks remain sensitive to tiny perturbations of the inputs. Indeed, the main issue still remains, that is, the training of such models results in decision boundaries that are too close to training and testing distributions. A perspective for future works is to investigate the possibility of leveraging the MAP estimation for the sake of a more robust training of the model, to improve the robustness of the model without affecting the accuracy.

The key takeaway of the Chapter 3 and Chapter 4 is that, despite the Lipschitz and UGNN models has to be considered a valid option for the achievement of certifiable robust accuracy in a safety-critical system, a bigger effort has to be pursued on improving the certifiable robust accuracy of such models. Indeed, the training of the models strongly relies on a loss function that increases the margin between training samples and the decision boundaries. Further techniques have to be investigated in future works to improve the training, such as augmentation or regularization strategies, and transfer-learning from larger datasets.

Acronyms

ALMA Augmented Lagrangian Method for Adversarial attacks.

AOL Almost Orthogonal Layer.

BCOP Block Convolution Orthogonal Parameterization.

CB Closest Boundary.

CLEVER Cross Lipschitz Extreme Value for nEtwork Robustness.

CW Carlini and Wagner.

DDN Decoupling Direction Norm.

DF DeepFool.

DNN Deep Neural Network.

ECO Explicitly Constructed Orthogonal.

FGSM Fast Gradient Sign.

FMN Fast Minimum Norm.

FOB Fast Outer Boundary.

IP Iterative Penalty.

L-BFGS-B Limited-memory Broyden–Fletcher–Goldfarb–Shanno Box-constrained.

MAP Minimal Adversarial Perturbation.

ONI Orthogonalization using Newton's Iteration.

RS Randomized Smoothing.

SDC Signed Distance Classifier.

SOC Skew Orthogonal Convolution.

UPD Unitary Pair Difference.

Bibliography

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *European Conference of Machine Learning and Knowledge Discovery in Databases*, 2013.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[3] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[4] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, 2018.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017.

[8] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *International Conference on Machine Learing (ICML)*, 2020.

[9] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] M. Pintor, F. Roli, W. Brendel, and B. Biggio, "Fast minimum-norm adversarial attacks through adaptive norm constraints," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[11] J. Rony, E. Granger, M. Pedersoli, and I. B. Ayed, "Augmented lagrangian adversarial attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[12] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *International conference on computer aided verification*, 2017.

[13] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learing (ICML)*, 2018.

[14] F. Brau, G. Rossolini, A. Biondi, and G. Buttazzo, "On the minimal adversarial perturbation for deep neural networks with provable estimation error," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023.

[15] J. Chen, J. Raghuram, J. Choi, X. Wu, Y. Liang, and S. Jha, "Revisiting adversarial robustness of classifiers with a reject option," in *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[17] B. Prach, F. Brau, G. Buttazzo, and C. H. Lampert, *1-lipschitz layers compared: Memory, speed, and certifiable robustness*, 2023. arXiv: 2311.16833.

[18] F. Brau, G. Rossolini, A. Biondi, and G. Buttazzo, "Robust-by-design classification via unitary-gradient neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, 2015.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] G. B. Folland, "Higher-order derivatives and taylor's formula in several variables," *Preprint*, pp. 1–4, 2005. [Online]. Available: https://sites.math.washington.edu/~folland/Math425/taylor2.pdf.

[22] D. Bertsekas, *Nonlinear Programming*, 2nd, ser. Athena scientific optimization and computation series. Athena Scientific, 1999.

[23] G. Rossolini, A. Biondi, and G. Buttazzo, "Increasing the confidence of deep neural networks by coverage analysis," *IEEE Transactions on Software Engineering*, 2022.

[24] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

[25] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses.," in *ICLR (Poster)*, 2018.

[26] F. Nesti, A. Biondi, and G. Buttazzo, "Detecting adversarial examples by input transformations, defense perturbations, and voting," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[27] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, 2016.

[28] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, ser. Athena scientific optimization and computation series. Athena Scientific, 1996, ISBN: 978-1-886529-04-5.

[29] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *Machine learning*, 2018.

[30] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "Boosting robustness certification of neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[31] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[32] A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, "Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks," 2019.

[33] C. Liu, R. Tomioka, and V. Cevher, "On certifying non-uniform bounds against adversarial attacks," in *International Conference on Machine Learing (ICML)*, 2019.

[34] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks.," in *UAI*, 2018.

[35] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[36] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," *International Conference on Machine Learing (ICML)*, 2018.

[37] R. H. Byrd, L. Peihuang, and J. Nocedal. (1996). "A limited-memory algorithm for bound-constrained optimization."

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[41] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," *Technical Report, Univeristy of Toronto*, 2009.

[42] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *International Conference on Learning Representations (ICLR)*, 2017.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[44] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learing (ICML)*, 2013.

[45] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, 2012.

[46] A. Wong, M. J. Shafiee, and M. S. Jules, "Micronnet: A highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification," *IEEE Access*, 2018.

[47] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax," *Journal of Open Source Software*, 2020.

[48] *Adversarial robustness toolbox*. [Online]. Available: https://github.com/Trusted-AI/adversarial-robustness-toolbox.git.

[49] L. Li, T. Xie, and B. Li, "Sok: Certified robustness for deep neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2023.

[50] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learing (ICML)*, 2019.

[51] N. Carlini, F. Tramer, K. D. Dvijotham, L. Rice, M. Sun, and J. Z. Kolter, "(Certified!!) adversarial robustness for free!," 2023.

[52] M. Losch, D. Stutz, B. Schiele, and M. Fritz, "Certified robust models with slack control and large lipschitz constants," 2023. arXiv: 2309.06166.

[53] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," *International Conference on Machine Learing (ICML)*, 2017.

[54] B. Prach and C. H. Lampert, "Almost-orthogonal layers for efficient general-purpose Lipschitz networks," in *European Conference on Computer Vision (ECCV)*, 2022.

[55] Q. Li, S. Haque, C. Anil, J. Lucas, R. B. Grosse, and J.-H. Jacobsen, "Preventing gradient attenuation in Lipschitz constrained convolutional networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[56] A. Trockman and J. Z. Kolter, "Orthogonalizing convolutional layers with the Cayley transform," *International Conference on Learning Representations (ICLR)*, 2021.

[57] K. Leino, Z. Wang, and M. Fredrikson, "Globally-robust neural networks," in *International Conference on Machine Learing (ICML)*, 2021.

[58] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," *International Conference on Machine Learing (ICML)*, 2019.

[59] S. Singla and S. Feizi, "Skew orthogonal convolutions," in *International Conference on Machine Learing (ICML)*, 2021.

[60] X. Xu, L. Li, and B. Li, "Lot: Layer-wise orthogonal training on improving l2 certified robustness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[61] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *International Conference on Learning Representations (ICLR)*, 2018.

[62] S Singla and S Feizi, "Fantastic four: Differentiable bounds on singular values of convolution layers," *International Conference on Learning Representations (ICLR)*, 2021.

[63] Å. Björck and C. Bowie, "An iterative algorithm for computing the best estimate of an orthogonal matrix," *SIAM Journal on Numerical Analysis*, 1971.

[64] L. Meunier, B. J. Delattre, A. Araujo, and A. Allauzen, "A dynamical system perspective for Lipschitz neural networks," in *International Conference on Machine Learing (ICML)*, 2022.

[65] F. Farnia, J. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," in *International Conference on Learning Representations (ICLR)*, 2018.

[66] T. Yu, J. Li, Y. Cai, and P. Li, "Constructing orthogonal convolutions in an explicit manner," *International Conference on Learning Representations (ICLR)*, 2021.

[67] R. Wang and I. Manchester, "Direct parameterization of Lipschitz-bounded deep networks," in *International Conference on Machine Learing (ICML)*, 2023.

[68] L. Huang, L. Liu, F. Zhu, D. Wan, Z. Yuan, B. Li, and L. Shao, "Controllable orthogonalization in training DNNs," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[69] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks," in *International Conference on Machine Learing (ICML)*, 2018.

[70] A. Cayley, "About the algebraic structure of the orthogonal group and the other classical groups in a field of characteristic zero or a prime characteristic.," *Journal für die reine und angewandte Mathematik*, 1846.

[71] E. Hoogeboom, V. Garcia Satorras, J. Tomczak, and M. Welling, "The convolution exponential and generalized Sylvester flows," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[72] A. Araujo, A. J. Havens, B. Delattre, A. Allauzen, and B. Hu, "A unified algebraic perspective on Lipschitz neural networks," in *International Conference on Learning Representations (ICLR)*, 2023.

[73] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[74] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, 2019.

[75] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, 2015.

[76] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020.

[77] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[78] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[79] S. Singla, S. Singla, and S. Feizi, "Improved deterministic l2 robustness on CIFAR-10 and CIFAR-100," in *International Conference on Learning Representations (ICLR)*, 2022.

[80] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," 2018.

[81] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. Del Barrio, "Achieving robustness in classification using optimal transport with hinge regularization," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[82] A. Chernodub and D. Nowicki, "Norm-preserving orthogonal permutation linear unit activation functions (OPLU)," in *International Conference of Artificial Neural Networks and Machine Learning (ICANN)*, 2016.

[83] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, *Provably minimally-distorted adversarial examples*, 2018. arXiv: 1709.10207.

[84] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[85] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," in *International Conference on Learning Representations (ICLR)*, 2018.

[86] S. Li, K. Jia, Y. Wen, T. Liu, and D. Tao, "Orthogonal deep neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[87] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[88] H. Federer, "Curvature measures," *Transactions of the American Mathematical Society*, 1959.

[89] S. Lang, *Fundamentals of differential geometry*. Springer Science & Business Media, 2012, vol. 191.

[90] T. Sakai, "On riemannian manifolds admitting a function whose gradient is of constant norm," *Kodai Mathematical Journal*, 1996.

[91] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[92] B. Prach and C. H. Lampert, "1-lipschitz neural networks are more expressive with n-activations," *arXiv preprint arXiv:2311.06103*, 2023.

[93] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, 1989.

[94] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[95] L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. Gonzalez Sanz, "Pay attention to your loss: Understanding misconceptions about lipschitz neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[96] E. M. Boczko and T. R. Young, *The signed distance function: A new tool for binary classification*, 2005. arXiv: `cs/0511105`.

[97] A. Abdollahpourrostam, M. Abroshan, and S.-M. Moosavi-Dezfooli, "Revisiting deepfool: Generalization and improvement," *arXiv preprint arXiv:2303.12481*, 2023.

[98] L. Béthune, P. Novello, T. Boissin, G. Coiffier, M. Serrurier, Q. Vincenot, and A. Troya-Galvis, *Robust one-class classification with signed distance function using 1-lipschitz neural networks*, 2023. [Online]. Available: `https://hal.science/hal-03977272` (visited on 02/13/2023).

[99] L. Béthune and M. Serrurier, "Certifiable metric one class learning with adversarially trained lipschitz classifier," in *NeurIPS ML Safety Workshop*, 2022.

Acknowledgment

A PhD is a long journey, and I am glad I had the chance to share it with so many interesting people. A huge thanks to Prof. Tommaso Cucinotta, who first guided me into the world of research, suggesting that I pursue a PhD, and whose contagious enthusiastic curiosity is a constant font of inspiration. An immense thanks to my supervisor, Prof. Giorgio Buttazzo, who guided me through the PhD, and whose patience and support were fundamental during the whole period. A sincere thanks to my tutor Prof. Alessandro Biondi, who supported me during the whole PhD, dwelling into the most technical details, and whose passion for the research and precision-oriented research perspective are a constant source of inspiration. A huge thanks to Prof. Battista Biggio, for the interesting conversations at the summer schools and the workshops, and for the support in finding a study abroad opportunity. An immense thanks to Prof. Christoph Lampert, who accepted me at the ISTA sharing his deep knowledge and expertise. Filippo, whose knowledge and interest in broader topics turns coffee pauses into a moment of learning and exploration. Giacomo, for the guidance during my period as a research fellow, and the constant support. Ara and Aromolo for the funny conversations and suggestions. Giulio for being a real brother-in-arm, sharing suggestions and ideas, and for being close either in the good or the bad times. Thanks to Federico and Edoardo for the interesting discussions. Thanks to Marco Pacini and Lorenzo De Marinins, for their inclusivity (e.k.a inceptionism attitude) and for being "the mental coach I wondered I had since my childhood[*]". Gianluca D'Amico, for supporting me in the battle against evil capitalism and knowledge-prostitution. Arman, for answering with incredible patience and dedication to all my dummy questions about physics. Zini, for reminding me that, at 12, there is no problem severe enough to break the call for mensa. Gabri Serra for reminding me that there is always enough time between an amaro and the last train. A huge thanks to Pietro Fara (Paolo[†]), whose attitude toward life is admirable, for the countless jams and funny moments. Thanks to Bernd for the nice moments we shared in Vienna, and for including me in nice projects either inside or outside the academy. Thanks to the new entries of the lab, Aristide, Giulia and Sania, for the funny moments and for bringing new and fresh research topics. An immense thanks to my childhood friends Andrea, Antonio, Daniele, Enrico, Fabio, Gabriele, Giacomo, Gianmaria, and Marco who know exactly zero of machine learning and DNNs, and hence, helped me the most reminding me that good life doesn't always require proofs. A special thanks go to my Brother, my Mother and my Father, for their presence and support in every moment.

[*]Gavazzi, 2021
[†]alias for Piero