université PARIS-SACLAY

Over-Approximating Neural Networks for Verification, Robustness, and Explainability

Sur-approximation de réseaux de neurones : applications à la vérification, la robustesse et les explications de réseaux de neurones

Thèse de doctorat de l'université Paris-Saclay

Ecole doctorale n°580, Sciences et Technologies de l'Information et de la Communication.
Spécialité de doctorat :informatique
Graduate School : Informatique et sciences du numérique.
Référent : ENS Paris-Saclay

Thèse préparée dans l'**Institut LIST (Université Paris-Saclay, CEA) et l'INRIA Paris** , sous la direction de **François TERRIER**, directeur de recherche, du co-encadrement de **Zakaria CHIHANI,** ingénieur - chercheur et du co-encadrement de **Caterina UBRAN**, chargée de recherche, INRIA - Paris / Ecole Normale Supérieure Paris / PSL

Thèse soutenue à Paris-Saclay, le 19 décembre 2025, par

Serge DURAND

Composition du Jury

Membres du jury avec voix délibérative

Pierre-Loïc GAROCHE Professeur, Ecole Nationale de l'Aviation Civile	Rapporteur & Examinateur
Mathieu SERRURIER Professeur, Institut de Recherche en Informatique de Toulouse	Rapporteur & Examinateur
Ileana OBER Professeure, Institut de Recherche en Informatique de Toulouse	Examinatrice
Claire PAGETTI Directrice de Recherche, ONERA	Examinatrice
Huan ZHANG Assistant Professor, University of Illinois Urbana-Champaign	Examinateur

THÈSE DE DOCTORAT

NNT : 2025UPASG101

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : Over-Approximating Neural Networks for Verification, Robustness, and Explainability

Keywords : Neural Networks, Verification, Robustness, Explainability, Certified Training

Abstract : The success of deep learning since the early 2010s has led to its adoption in a wide range of applications, from image classification to natural language processing. However, deploying AI models — especially in safety-critical applications — requires strong behavioral guarantees as well as means to understand and explain their predictions. Robustness — the ability of a model to maintain performance under small input changes — is an important safety requirement and helps in formally explaining model predictions.

Over-approximation techniques, introduced by the machine learning and formal methods communities, have emerged as a key tool to verify neural networks and to train them with provable robustness goals. By bounding the range of possible network outputs, they make it possible to prove desirable properties for all possible inputs within a specified input domain. In this thesis we investigate how over-approximations can be effectively exploited to further improve the verification, training, and formal explainability of neural networks.

Such over-approximations are essential to make the analysis tractable for neural networks of moderate scale, but they also tend to be imprecise, in particular for networks trained with no robustness goals. In a first contribution we leverage key information from over-approximations to efficiently tighten bounds when verifying trained networks with low-dimensional input spaces. This technique is implemented in PyRAT, a neural network verification tool, and contributed to its success in several benchmarks of the latest edition of the International Verification of Neural Networks Competition (VNN-COMP 2024).

A second contribution shows that certified training - using over-approximations to minimize a sound bound on the worst-case loss - can be mixed with adversarial training - computing the loss over perturbed training samples - to improve empirical robustness and prevent catastrophic overfitting, a failure mode of single-step adversarial training, under specific experimental settings.

Finally, we propose to use over-approximations to train for formal explainability. To avoid the trivial case where robustness voids all explanations, we introduce Feature Subset Certified Training, a new scheme that enforces robustness only over selected subsets of input features, a first step towards better trade-offs between accuracy and conciseness of the explanations.

Together, these contributions illustrate how over-approximations can be leveraged towards better robustness and explainability, supporting the development of safer and more trustworthy AI systems.

Titre : Sur-approximation de réseaux de neurones : applications à la vérification, la robustesse et les explications de réseaux de neurones.

Mots clés : Réseaux de Neurones, Vérification, Robustesse, Explicabilité, Entraînement Robuste

Résumé : Depuis le début des années 2010, le succès de l'apprentissage profond a conduit à son adoption dans un large éventail d'applications, allant de la classification d'images au traitement du langage naturel. Cependant, le déploiement de modèles d'IA — en particulier dans des systèmes critiques — nécessite à la fois de fortes garanties et des moyens de comprendre et d'expliquer leurs prédictions. La robustesse — la capacité d'un modèle à maintenir ses performances face à de petites perturbations de ses entrées — constitue un prérequis important en matière de sécurité et contribue à l'explicabilité formelle des prédictions du modèle.

Les techniques de sur-approximation, introduites par les communautés de l'apprentissage automatique et des méthodes formelles, se sont imposées comme un outil clé pour vérifier les réseaux de neurones et les entraîner avec des objectifs de robustesse prouvable. En bornant l'ensemble des sorties possibles d'un réseau, elles permettent de prouver certaines propriétés souhaitées pour toutes les entrées appartenant à un domaine donné. Dans cette thèse, nous étudions dans quelle mesure ces sur-approximations peuvent être utilisées de manière efficace pour améliorer la vérification, l'apprentissage et l'explicabilité formelle des réseaux de neurones.

De telles sur-approximations sont essentielles pour rendre possible l'analyse de réseaux de neurones de taille réaliste, mais elles ont également tendance à être imprécises, en particulier pour des réseaux entraînés sans objectif explicite de robustesse. Dans une première contribution, nous exploitons des informations clés issues des sur-approximations pour affiner efficacement les bornes lors de la vérification de réseaux avec des espaces d'entrée de faible dimension. Cette technique est implémentée dans PyRAT, un outil de vérification de réseaux de neurones, et a contribué à ses bons résultats dans la dernière édition de la compétition internationale de vérification de réseaux de neurones (VNN-COMP 2024).

Une seconde contribution montre que l'entraînement certifié — qui utilise des sur-approximations pour minimiser une borne de la fonction de coût dans le pire cas — peut être combiné à l'entraînement adversarial — qui calcule la fonction de coût sur des échantillons d'entraînement perturbés — afin d'améliorer la robustesse empirique et de prévenir le catastrophic overfitting, un mode d'échec de l'entraînement adversarial, dans certains contextes expérimentaux.

Enfin, nous proposons d'utiliser les sur-approximations pour entraîner des modèles en vue d'une explicabilité formelle. Afin d'éviter le cas trivial où la robustesse rend toute explication vide, nous introduisons le Feature Subset Certified Training, un nouveau mode d'entraînement qui impose la robustesse uniquement sur certains sous-ensembles des features des échantillons, un premier pas vers de meilleurs compromis entre précision et concision des explications formelles.

Dans l'ensemble, ces contributions illustrent comment les sur-approximations peuvent être mises à profit pour améliorer la robustesse et l'explicabilité, soutenant ainsi le développement de systèmes d'IA plus sûrs et plus dignes de confiance.

Contents

List of Figures

List of Tables

1 Introduction

Deep learning has seen great success in the past decades [9–12]. The rise of Large Language Models (LLMs) has brought the capabilities of modern artificial intelligence (AI) systems to the general public, leading many to become active users. We focus in this thesis on **connectionist** AI, or machine learning, which relies on data to train models, as opposed to **symbolic** AI, which relies on explicit knowledge and handcrafted rules. We are interested in particular in the family of models known as neural networks. The success of neural networks has not followed a straightforward trajectory. After the pioneering work on the perceptron by Rosenblatt in the late 1950s [13], the field of machine learning (ML) entered a period of reduced interest, commonly referred to as the "AI winter". A series of breakthroughs, including the development of the efficient backpropagation algorithm [14–16], the introduction of convolutional neural networks [17] and their training by backpropagation [18, 19], the advent of Graphical Processing Units (GPUs) for training [20], and the availability of large datasets such as ImageNet [21], paved the way for the current deep learning revolution.

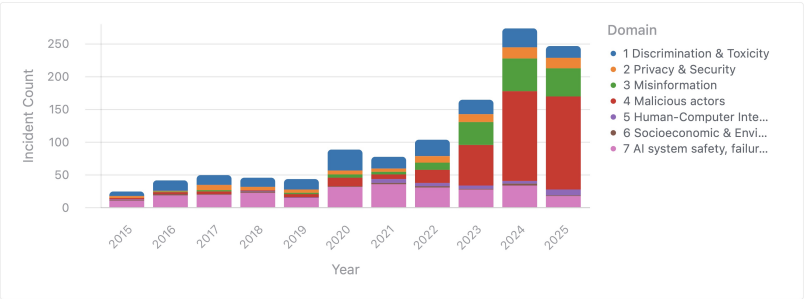


Figure 1.1: Incidents involving AI systems since 2015. The chart comes from `https://airisk.mit.edu/ai-incident-tracker/timeline-risk-classification`, the data from `https://incidentdatabase.ai/`. Completing the legend: 5: Human-Computer Interaction, 6: Socioeconomic and Environmental, 7: AI system safety, failures and limitations.

As AI systems are increasingly adopted in real-world applications, including safety-critical domains such as autonomous driving [22] and medical diagnosis [23], the need for reliable and trustworthy AI has become paramount. AI-related incidents have risen sharply in recent years, particularly those involving malicious actors, as illustrated in Figure 1.1. Such incidents range from biased decision-making [24] and safety failures in autonomous systems [25] to malicious uses of AI, such as deepfakes in automated disinformation campaigns [26]. Governments and regulatory bodies are increasingly recognizing the need for trustworthy AI, as exemplified by the European Union's AI Act [27]. Ensuring the reliability of AI systems requires rigorous methods to verify their behavior, a challenge that has long been central in the broader field of software verification.

1.1 Over-Approximations and Trustworthy AI

1.1.1 Verification of Software Systems

The need for reliable and trustworthy systems is not unique to AI. We briefly discuss the field of software verification, as it has inspired many of the techniques used in building trustworthy AI systems. On a more

[9]: Krizhevsky et al. (2012), 'ImageNet Classification with Deep Convolutional Neural Networks'

[10]: Silver et al. (2016), 'Mastering the game of Go with deep neural networks and tree search'

[11]: Brown et al. (2020), 'Language Models are Few-Shot Learners'

[12]: Jumper et al. (2021), 'Highly accurate protein structure prediction with AlphaFold'

[13]: Rosenblatt (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.'

[14]: Linnainmaa (1976), 'Taylor expansion of the accumulated rounding error'

[15]: Werbos (1982), 'Applications of advances in nonlinear sensitivity analysis'

[16]: Rumelhart et al. (1986), 'Learning representations by back-propagating errors'

[17]: Fukushima (1980), 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'

personal note, formal verification has historically been the focus of both laboratories I have been affiliated with during my thesis: the ANTIQUE team at Inria Paris / École normale supérieure (ENS) and the *Laboratoire de Sûreté et de Sécurité du Logiciel* (LSL) at CEA LIST. Both have developed major industrial tools for software verification: *Frama-C* [28], developed at LSL, and *Astrée* [29], developed at ANTIQUE. Software verification has been an active field of research for decades, with the goal of ensuring that programs behave as intended and are free from bugs and vulnerabilities. Formal methods, such as model checking [30], theorem proving [31, 32], and abstract interpretation [33] have been developed to provide mathematical guarantees about the correctness of software systems. Such methods are crucial for ensuring the reliability of safety-critical systems, where testing alone is insufficient. The infamous Ariane 5 launch failure in 1996 cost 370 million US dollars and was caused by a software error undetected during testing[1]. Beyond financial losses, undetected software problems can lead to fatalities: the Therac-25 radiation therapy machine caused several deaths in the 1980s due to software errors [34]. These incidents highlight the importance of formal verification methods for ensuring the safety and reliability of software systems.

Automated software verification faces a fundamental theoretical limitation: it cannot simultaneously be **complete** (able to verify all correct programs), **sound** (never producing false negatives—that is, if the verifier claims a program is correct, it actually is), and **fully automated**. This impossibility is a consequence of Rice's theorem [35] about the undecidability of non-trivial software properties[2]. Abstract Interpretation in particular trades completeness for soundness and automation by over-approximating program behaviors, *e.g.* for static analysis. Using abstract domains and transfer functions defined for them one can analyze properties of a program. For example when analyzing numerical properties of a program, *e.g.* the absence of overflow, an **abstract domain** (such as intervals) and **transfer functions** can be used to compute sound over-approximations of the possible values of the variables at different points in the program and prove the absence of overflow, without actually executing the program on every possible input.

1.1.2 Over-Approximations of Neural Networks

In the context of neural networks, **over-approximations** refer to a family of techniques designed to bound the range of possible outputs of a model, given a range of possible inputs. This is done by propagating **sound** bounds on the inputs through the network: although the exact reachable space at each layer may not be known, it is always contained within the propagated bounds. As a result, the bounds on the outputs encompass all possible outputs for inputs within the specified range. These over-approximated bounds can then be used to demonstrate that the network's decision is stable within the input range. However, if the bounds are too loose, the verification process may be inconclusive. Because bound propagation is typically differentiable, it is possible to incorporate bounds into the training process. This approach is known as **certified training**, or **provably robust training**, as it enforces a sound verifiable robustness guarantee on the trained network. A schematic representation of the idea is given in Figure 1.2. On the other hand, **adversarial training** [37] relies on finding specific perturbations of the training data that fool the network and training the network to be robust to these perturbations.

[18]: Zhang et al. (1990), 'Parallel distributed processing model with local space-invariant interconnections and its optical architecture'

[19]: LeCun et al. (1989), 'Backpropagation Applied to Handwritten Zip Code Recognition'

[20]: Raina et al. (2009), 'Large-scale deep unsupervised learning using graphics processors'

[21]: Deng et al. (2009), 'Imagenet: A large-scale hierarchical image database'

[22]: Zhao et al. (2025), 'A Survey of Autonomous Driving from a Deep Learning Perspective'

[23]: Ahsan et al. (2022), 'Machine-learning-based disease diagnosis: A comprehensive review'

[24]: Angwin et al. (2016), *Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks*

[25]: Wikipedia contributors (2024), *List of Tesla Autopilot crashes*

[26]: Zellers et al. (2019), 'Defending against neural fake news'

[27]: European Union (2024), *Regulation (EU) 2024/1689 Artificial Intelligence Act*

[28]: Kirchner et al. (2015), 'Frama-C: A software analysis perspective'

[29]: Blanchet et al. (2002), 'Design and Implementation of a Special-Purpose Static Program Analyzer for Safety-Critical Real-Time Embedded Software'

[30]: Clarke and Emerson (1982), 'Design and synthesis of synchronization skeletons using branching time temporal logic'

[31]: Coquand and Huet (1986), *The calculus of constructions*

[32]: Paulson (1989), 'The foundation of a generic theorem prover'

[33]: Cousot and Cousot (1977), 'Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints'

1: https://esamultimedia.esa.int/docs/esa-x-1819eng.pdf

[34]: Leveson and Turner (1993), 'An investigation of the Therac-25 accidents'

[35]: Rice (1953), 'Classes of recursively enumerable sets and their decision problems'

2: Non-trivial properties are properties that are neither true nor false for every program.

Abstract interpretation is not restricted to the analysis of numerical properties. It is underpinned by a rich mathematical theory based on lattice theory. We refer to Cousot [36] for more details.

[37]: Madry et al. (2018), 'Towards Deep Learning Models Resistant to Adversarial Attacks'

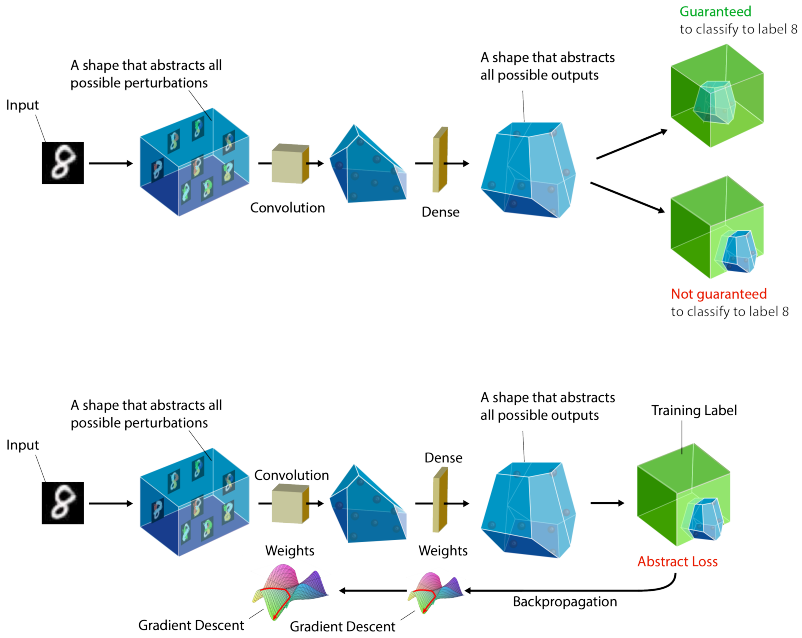


Figure 1.2: Over-approximating neural networks outputs for verification and training. Credit: ETH Zurich SRI Lab.

Related terminology The term **bound propagation** is often used to describe the techniques underlying neural network over-approximation: the range of outputs is derived by propagating bounds on the inputs through the network. **Reachability analysis** also relates to the idea: the goal is to compute the set of possible outputs (the reachable set) from a given set of inputs. We note however that while over-approximations can be used for reachability analysis the term also includes techniques for exact reachable set computation. **Linear approximations** or **linear relaxations** are also central for neural network over-approximation. Efficient and precise bound propagation requires approximating the non-linear behavior of neural networks with linear functions. Finally, the term **abstract domains**, or simply **abstractions**, from abstract interpretation theory, are also used to refer to some implementations of over-approximations from this perspective.

Remark 1.1.1 Linear approximations (or relaxations) can be used to propagate lower and upper bounds through the network directly using its weights, but also to model the bounding problem in a suitable formulation for off-the-shelf solvers, such as Linear Programming (LP) solvers. In this thesis we focus on the former use of linear approximations.

Brief history These techniques have been developed by both the machine learning and formal methods communities, often concurrently, starting in 2017 and 2018. From an abstract interpretation perspective, notable works include [38] with the AI2 tool, [39] with *DeepZ* (a zonotope abstraction), and [40] with *DeepPoly* (a polyhedra abstraction), both adapted to neural networks and implemented in the ERAN tool[3] and its successor, MN-BAB[4] [41]. From the machine learning perspective, [42] and [43] used equivalent linear approximations for bound propagation[5]. Later that same year, [44] extended their approach to more architectures, including networks with skip connections and general

[38]: Gehr et al. (2018), 'AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation'

[39]: Singh et al. (2018), 'Fast and Effective Robustness Certification'

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

3: `https://github.com/eth-sri/eran`

4: `https://github.com/eth-sri/mn-bab`

[41]: Ferrari et al. (2022), 'Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound'

[42]: Wong and Kolter (2018), 'Provable defenses against adversarial examples via the convex outer adversarial polytope'

[43]: Weng et al. (2018), 'Towards fast computation of certified robustness for ReLU networks'

5: In Wong and Kolter [42], the goal is to train robust networks, while Weng et al. [43] aims to compute the largest input range on which a trained network is robust

[44]: Wong et al. (2018), 'Scaling provable adversarial defenses'

non-linear activation functions. Concurrently, [45] introduced CROWN, which uses different linear (and quadratic) approximations for general architectures. CROWN is the core abstraction used in Auto_LiRPA[6] [46], a popular library for differentiable bound propagation used for both training and verification, particularly in α,β-CROWN[7], a state-of-the-art neural network verifier. We refer to [47] for a survey of algorithms for neural networks, not restrained to over-approximations, [48] for a pedagogical introduction, and [49] for a higher level perspectives on the use of formal methods for trustworthy AI.

We introduce in chapter 2 the background and formal definitions on neural networks, their robustness both from an empirical and from a verified perspective. We also discuss the state of the art in certified training, and the technical aspects of linear approximations for bound propagation

[45]: Zhang et al. (2018), 'Efficient Neural Network Robustness Certification with General Activation Functions'

6: https://github.com/Verified-Intelligence/auto_LiRPA

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

7: https://github.com/Verified-Intelligence/alpha-beta-CROWN

[47]: Liu et al. (2019), 'Algorithms for Verifying Deep Neural Networks'

[48]: Albarghouthi (2021), 'Introduction to Neural Network Verification'

[49]: Urban and Miné (2021), 'A Review of Formal Methods applied to Machine Learning'

1.1.3 Robustness of Neural Networks

A central property of neural networks studied in this thesis is their **local robustness**. In simple terms, a neural network is locally robust if its decision does not change within a small neighborhood around a given input. Neural networks have been shown to be vulnerable to small, carefully crafted perturbations of their inputs, the so-called **adversarial examples** [1, 50, 51]. An example is shown in Figure 1.3. Robustness can be also be defined with respect to semantically meaningful perturbations, such as rotations, or against distribution shifts. Distribution shifts might happen when a network is trained on data from one distribution (*e.g.* images taken in daylight) and deployed on data from another (*e.g.*, images taken at night). Robustness can also be defined **globally**, particularly for networks with low-dimensional input spaces and well-defined input domains (such as ranges of speed and altitude for an aircraft collision avoidance system). Beyond robustness to input perturbations, other important properties include fairness [52], explainability [53], and privacy [54]. Local robustness is a recurrent notion in both the definition and the verification of properties beyond the local adversarial setting. It is mathematically well-defined, and many tools have been developed to promote or assess it. Many works studying properties beyond local robustness build on techniques originally developed for local robustness. For instance, [55] prepend layers to the network to map semantically meaningful perturbations (such as rotations) to ℓ_p-norm–bounded perturbations, and then apply local robustness verification techniques. Even in the case of distribution shifts, robustness can be characterized in terms of local robustness by exploiting generative models and applying perturbations in a latent space rather than the input space [56]. [57] employ various verification techniques developed for robustness to reduce the search space in fairness verification.

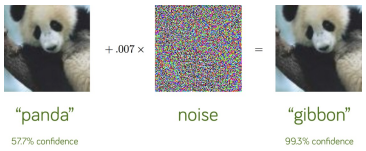


Figure 1.3: An adversarial example for an image classifier. A small perturbation (amplified for visualization purposes) is added to the original image of a panda, leading the network to misclassify it as a gibbon with high confidence. Figure from Goodfellow et al. [1].

[1]: Goodfellow et al. (2015), 'Explaining and Harnessing Adversarial Examples'

[50]: Biggio et al. (2013), 'Evasion attacks against machine learning at test time'

[51]: Szegedy et al. (2014), 'Intriguing properties of neural networks'

[52]: Urban et al. (2019), 'Perfectly Parallel Fairness Certification of Neural Networks'

[53]: Marques-Silva and Ignatiev (2022), 'Delivering Trustworthy AI through Formal XAI'

[54]: Lecuyer et al. (2019), 'Certified robustness to adversarial examples with differential privacy'

1.2 Tightening Bounds for Verification

Despite their usefulness, over-approximations face a fundamental trade-off between precision and computational tractability. Tight approximations are necessary to prove meaningful properties, but they quickly become computationally prohibitive for networks of large scale. Conversely, efficient over-approximations often produce bounds that are too loose to verify properties of interest, especially for networks trained

without explicit robustness objectives. This precision-scalability trade-off is a central challenge in making formal methods practical for real-world neural networks.

In Part 'Tightening Bounds for Incomplete Verification', we focus on verifying relatively small neural networks with low-dimensional inputs, where standard over-approximation techniques are tractable but too loose to establish the desired properties. We begin by describing a simple yet powerful technique to improve the precision of over-approximations by partitioning the input space and verifying each part separately, introduced in [58]. We then discuss different strategies to choose the input dimensions on which to partition, and propose a new heuristic to prioritize the subproblems to solve first.

[55]: Mohapatra et al. (2020), 'Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations'

[56]: Wu et al. (2022), 'Toward Certified Robustness Against Real-World Distribution Shifts'

[57]: Mazzucato and Urban (2021), 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

Contributions In Chapter 3, we propose a new prioritizing heuristic, ReCIPH (*Relational Coefficient for an Input Partitioning Heuristic*). We rely on the linear approximations, already computed to propagate bounds, to rank the inputs and chose the one on which to partition first. Implemented in PyRAT [59], we evaluate it on a standard benchmark of small networks with low-dimensional inputs and a use-case from an industrial partner, showing the improvement over existing heuristics applied to networks with different activation functions. We also include results in the context of fairness verification with Libra [57] and report the results of the relevant benchmarks of the International Verification of Neural Networks Competition (VNN-COMP 2024) [60].

[59]: Lemesle et al. (2024), 'Neural Network Verification with PyRAT'

[60]: Brix et al. (2024), 'The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results'

1.3 Over-approximations beyond verified robustness

Although over-approximations have been primarily developed to verify or enforce the provable robustness of neural networks, they can be employed in other applications. Part 'Applications of Certified Training' focuses on two applications beyond verified robustness: empirical robustness and formal explainability. For both applications, we focus on using certified training during the training process. We argue that integrating desirable properties into the training process is crucial for obtaining scalable solutions, as already observed in the case of verified robustness.

1.3.1 Empirical Robustness

Combining adversarial training with certified training has been explored in the literature and currently represents the state of the art for training verifiably robust networks with **Expressive Losses** [8]. However, their combination for empirical robustness alone remains underexplored. Adversarial training requires computing perturbations of the original samples of the training data [37]. These perturbations are computed using multiple forward and backward passes through the network, which results in a high computational cost. To reduce this cost, single-step adversarial training has been proposed, where only one perturbation step is computed [6, 61]. However, single-step adversarial training suffers from a failure mode known as catastrophic overfitting [6]: after several training epochs, the network becomes vulnerable to multi-step attacks while still appearing robust to single-step attacks. In Chapter 4 we investigate

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[37]: Madry et al. (2018), 'Towards Deep Learning Models Resistant to Adversarial Attacks'

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

[61]: Shafahi et al. (2019), 'Adversarial training for free!'

the use of expressive losses to both prevent catastrophic overfitting and bridge the gap toward multi-step adversarial training.

Contributions In Chapter 4, we show that combining certified training with single-step adversarial training, when carefully tuned, can prevent catastrophic overfitting across several datasets and architectures. We also show that on other settings it can bridge the gap between single-step and multi-step adversarial training. These contributions have been published in the Transactions on Machine Learning Research (TMLR) journal [62].

[62]: De Palma et al. (2025), 'On Using Certified Training towards Empirical Robustness'

1.3.2 Explainability of Neural Networks

Another important aspect of trustworthy AI systems is their **explainability**. We say that a model is explainable when a human user can understand and interpret the decisions made by AI models. Explainability depends on a model and on the choice of what constitutes an explanation.[8] In this thesis, we focus in particular on explanations defined as a measure of importance of the input features for the model's decision. Most of the existing techniques to estimate feature importance are heuristic-based and relies on gradients [65–67], perturbations [68] or training surrogate interpretable models [69].

The need for explainability is often motivated by the so-called *black box* nature of neural networks. We prefer to attribute it instead to the scale of modern networks. When model architectures and weights are available to the user, the model is not truly a black box, but its decisions become opaque due to their complexity. This is not unique to neural networks: decision trees, typically considered interpretable, can also become opaque when large. In fact, neural networks can be transformed into decision trees [63], but the resulting trees are typically too large to interpret.

[63]: Aytekin (2022), 'Neural networks are decision trees'

8: We refer to Xu-Darme [64] for a discussion and possible disambiguation of the terms *explainability*, *interpretability*, *transparency*.

[64]: Xu-Darme (2023), 'Algorithms and evaluation metrics for improving trust in machine learning : application to visual object recognition'

[65]: Simonyan et al. (2014), *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

[66]: Sundararajan et al. (2017), 'Axiomatic attribution for deep networks'

[67]: Smilkov et al. (2017), *SmoothGrad: removing noise by adding noise*

[68]: Binder et al. (2016), 'Layer-wise relevance propagation for neural networks with local renormalization layers'

[69]: Ribeiro et al. (2016), '" Why should i trust you?" Explaining the predictions of any classifier'

[70]: Huang and Marques-Silva (2023), 'From Robustness to Explainability and Back Again'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[72]: Bassan and Katz (2023), 'Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks'

A recent line of work [70–72] seeks to go beyond heuristic approaches to estimating feature importance and instead aims at **Formal Explanations**. In this framework, input features are **irrelevant** if they *provably* do not change the model's decision, and **relevant** otherwise. Existing algorithms for computing irrelevant features in neural networks involve ordering the input features and querying a verifier for each dimension, which yields the maximal set of irrelevant features for that ordering. This approach relies heavily on neural network verification and faces serious scalability issues due to repeated verification queries. Current attempts to improve the scalability of formal explanations focus on algorithmic enhancements applied post-training. These approaches often involve a relaxed notion of formal explanations: the irrelevant features remain verifiably irrelevant, but the set is no longer guaranteed to be maximal. To make formal explainability more practical, in Chapter 5 we investigate how certified training can be used to train networks that are easier to explain formally.

Contributions In Chapter 5 we make contributions toward scaling formal explainability. We introduce the notion of empirically ϵ-formally explainable models, which quantifies the explainability of networks in the context of relaxed formal explanations. Inspired by the partitioning heuristic of Chapter 3, we propose a new ordering heuristic. We show that, on networks trained with robustness objectives, it systematically leads to larger sets of irrelevant features, *i.e.* more concise explanations, compared to existing heuristics. We introduce training method, Feature Subset Certified Training (FSCT). This method relies on certified training to promote robustness only on subsets of each input, aligning with the desiderata of formal explanations. We compare FSCT with existing certified and adversarial training approaches, highlighting the trade-offs on CIFAR-10 and TinyImageNet. In some settings (high perturbation budget on TinyImageNet) FSCT provides the best trade-off between accuracy, explainability and average size of explanations.

Each chapter of the thesis concludes with a related work section, providing a literature review on the relevant topics. The thesis ends with a concluding chapter that summarizes the contributions and discusses future research directions.

Notations

$\mathbb{R}$	Set of real numbers.
$\mathbb{R}^d$	d-dimensional real space.
$\mathbb{N}$	Set of natural numbers.
$\boldsymbol{x}$	Vectors are in bold italic lowercase letters.
$\mathbf{1}$	Vector of ones.
$\mathbf{0}$	Vector of zeros.
x	Scalars are in italic lowercase letters.
$\boldsymbol{x}[i]$	i-th component of a vector $\boldsymbol{x}$.
$\boldsymbol{M}$	Matrices are in bold italic upperca+e letters.
$\|\boldsymbol{x}\|$	Norm of the vector $\boldsymbol{x}$ (default: ℓ_∞ norm).
$B(\boldsymbol{x}, \epsilon)$	ϵ-ball centered at $\boldsymbol{x}$ induced by a norm: $B(\boldsymbol{x}, \epsilon) := \{x' \mid \|x - x'\| < \epsilon\}$.
$[\![a, b]\!]$	Discrete interval of integers $\{a, a+1, \ldots, b\}$ for $a, b \in \mathbb{N}$ and $a < b$.
$[\![n]\!]$	Abuse of above notation for integers $\{0, \ldots, n-1\}$ for $n \in \mathbb{N}$.
$\mathcal{S}, \mathcal{Y}, \mathcal{D}$	Generic sets are in uppercase calligraphic letters.
$\boldsymbol{x}[\mathcal{S}]$	Subvector of $\boldsymbol{x} \in \mathbb{R}^d$ indexed by $\mathcal{S} \subseteq [\![d]\!]$; if $\|\mathcal{S}\| = d'$, then $\boldsymbol{x}[\mathcal{S}] \in \mathbb{R}^{d'}$.
$\mathcal{I}^c$	Complement of a set $\mathcal{I} \subseteq \mathcal{U}$: $\{i \in \mathcal{U} \mid i \notin \mathcal{I}\}$ (*e.g.*, $\mathcal{U} = [\![d]\!]$).
f_θ	Neural network with parameters $\boldsymbol{\theta}$.

Background and Context

Background 2

We first present the necessary background on neural networks, their training and the problem of their robustness. We introduce some of the classical methods to train them in a robust manner and to verify formally their robustness. We focus on bound propagation methods, in particular Interval Bound Propagation (IBP), central to the training of robust models, and linear approximations, used in verification.

2.1 Supervised Classification and Neural Networks

We focus in this thesis on the problem of supervised classification. Let $\mathcal{X}$ be the input space, for example $\mathcal{X} = \mathbb{R}^d$, and $\mathcal{Y}$ be the output space, for example $\mathcal{Y} = [\![K]\!]$ for a multiclass classification problem with K classes.

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$, with n **data points** the goal of supervised classification is to find a function $f : \mathcal{X} \to \mathbb{R}^K$ such that $\operatorname{argmax}_{k \in [\![K]\!]} f(\boldsymbol{x}^{(i)})[k] = y^{(i)}$ for all samples $\boldsymbol{x}^{(i)}$, with $f \in \mathcal{F}$ some family of functions, called **hypothesis class**. In this thesis we focus on the class of neural networks.

2.1.1 Neural Networks

A popular family of functions used to solve the classification problem are **neural networks**, particularly used for image classification since the success of AlexNet [9] in 2012.

In their simplest form neural networks can be seen as the composition of functions, alternating between linear and non-linear functions, applied sequentially to the input.

Definition 2.1.1 (Neural Network) *A neural network f_θ parametrized by θ is a function $f_\theta : \mathcal{X} \to \mathbb{R}^K$ that can be expressed as a composition of L functions $f_1, \dots, f_L$ such that:*

$$f_\theta(\boldsymbol{x}) = \left(f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1\right)(\boldsymbol{x}) \tag{2.1}$$

with each function $f_l, l \in \{1, \dots L\}$ defined as:

$$f_l(\boldsymbol{x}) = \sigma\left(\phi_l(\boldsymbol{x})\right) \tag{2.2}$$

where σ is a non-linear **activation function**[1] and ϕ_l a linear function. Common activation functions include:

- **ReLU** (Rectified Linear Unit) $\sigma(x) = \max(0, x)$
- **sigmoid**: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$
- **tanh**: $\sigma(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$.

$\operatorname{argmax}_{k \in [\![K]\!]} \boldsymbol{x}[k]$ is a short notation for the index of the component of $\boldsymbol{x} \in \mathbb{R}^K$ with maximum value $\operatorname{argmax}_{k \in [\![n]\!]} \{\boldsymbol{x}[k] : k \in \{0, \dots n\}\}$ with $\boldsymbol{x}$ some vector of $\mathbb{R}^K$.

[9]: Krizhevsky et al. (2012), 'ImageNet Classification with Deep Convolutional Neural Networks'

1: For simplicity, we abuse notations and use σ to denote both the activation function defined as real functions: $\sigma : \mathbb{R} \to \mathbb{R}$, and its elementwise extension to vectors $\boldsymbol{x} \in \mathbb{R}^d$, obtained by applying it to each coordinate.

Example 2.1.1 (Multi-layer Perceptron) In the case of a **multi-layer perceptron** (MLP), also called **Fully Connected Network** (FCN), the functions are simply affine transformations: $\phi_l(\boldsymbol{x}) = \boldsymbol{W}_l\boldsymbol{x} + \boldsymbol{b}_l$, where $\boldsymbol{W}_l$ is a weight matrix and $\boldsymbol{b}_l$ is a bias vector of appropriate dimensions. In this case the parameters θ are the weights and biases of the network: $\theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$.

The raw outputs of the network $f_\theta(\boldsymbol{x})$ are called **logits**.

2.1.2 Neural Network Training

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, the goal of neural network training is to find the parameters θ minimizing the empirical risk:

$$\min_\theta \sum_{i=1}^{n} \mathcal{L}\left(f_\theta(\boldsymbol{x}^{(i)}), y^{(i)}\right) \tag{2.3}$$

with $\mathcal{L}$ a loss function, typically the cross-entropy loss for multiclass classification. While the problem is non-convex due to the use of non-linear activation functions it can be solved approximately using gradient descent methods.

Definition 2.1.2 (Cross-entropy) *The cross-entropy loss is defined as*

$$\mathcal{L}(\boldsymbol{x}, y) = -\log\left(\frac{e^{\boldsymbol{x}[y]}}{\sum_{k=1}^{K} e^{\boldsymbol{x}[k]}}\right)$$

for a vector $\boldsymbol{x} \in \mathbb{R}^K$ and a label $y \in [\![k]\!]$.

Remark 2.1.1 Loss functions are central to machine learning. They can also be called **cost functions** or **objective functions**. They can be defined on one-hot encoded labels, so that y has the same dimension as $\boldsymbol{x}$, they can also expect probabilities as inputs, in which case the softmax function is applied to the logits before computing the loss. In practice, we use the cross-entropy loss expecting logits and integer labels as the training frameworks (PyTorch, TensorFlow . . .) combine the softmax and cross-entropy in a single function for numerical stability.

For a differentiable function $\varphi : \mathbb{R}^d \to \mathbb{R}$, we denote its gradient by $\nabla\varphi$. Moreover, if θ is some set of parameters involved in the computation of ϕ, we denote by $\nabla_\theta\phi$ the gradient of ϕ with respect to the parameters θ[2]. Similarly, we denote by $\nabla_{\boldsymbol{x}}\phi \in \mathbb{R}^d$ the gradient of ϕ with respect to its input $\boldsymbol{x}$. The gradient of the loss function $\mathcal{L}$ with respect to the parameters θ can be computed efficiently using **backpropagation** [14, 17, 32], an application of the chain rule of differentiation. The parameters θ are updated iteratively using the gradient of the loss function:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta\mathcal{L}\left(f_\theta(\boldsymbol{x}^{(i)}; \theta_t), y^{(i)}\right) \tag{2.4}$$

with η the learning rate, controlling the magnitude of the update, and t the iteration index. The training is typically done by processing mini-batches of the dataset randomly sampled (**Stochastic Gradient Descent**). One **epoch** is defined as a full pass over the dataset, *i.e.*, processing all samples once.

Training in practice Although we mentioned that gradient descent can *approximately* solve the empirical risk minimization problem Eq. (2.3), there is no guarantee that it will find a good solution for most non-convex problems. Convergence analysis is an active area of research [73, 74] and theoretical results applies to specific settings. In practice the distribution of the data is unknown, and the machine learning practitioner has to experimentally find a good set of hyperparameters (learning rate, batch size, architecture, etc.) to obtain a model that has good performance and generalizes well on unseen data. Crucially, the dataset $\mathcal{D}$ is randomly split into a training set, used to train the model, and a test set, used to evaluate the performance of the trained model on unseen data. The performance is typically measured as the accuracy on the test set: the proportion of samples correctly classified by the model. When iterating over hyperparameters, a third subset, called the validation set, is used to avoid overfitting the test set.

2: A careful reader will wonder about networks with ReLU activations: they are not differentiable on their entire input space as the ReLU function is not differentiable in 0. However, they are differentiable almost-everywhere: the non-differentiable inputs have measure 0 in the input space. In practice ReLU activations are commonly used in training neural networks.

[14]: Linnainmaa (1976), 'Taylor expansion of the accumulated rounding error'

[17]: Fukushima (1980), 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'

[32]: Paulson (1989), 'The foundation of a generic theorem prover'

[73]: Arora et al. (2019), 'A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks'

[74]: Zhang et al. (2019), 'Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks'

We did not go into details about the modern architectures of neural networks (convolutions [17–19], residual connections [75] ...) or the training techniques (batch normalization [76], dropout [77]...) as they are not central to the contributions of this thesis. We refer the interested reader to Murphy [78] for a comprehensive introduction to deep learning. We will state the standard architectures and training techniques used in our experiments in the corresponding sections.

[18]: Zhang et al. (1990), 'Parallel distributed processing model with local space-invariant interconnections and its optical architecture'

[19]: LeCun et al. (1989), 'Backpropagation Applied to Handwritten Zip Code Recognition'

[75]: He et al. (2016), 'Identity mappings in deep residual networks'

[76]: Ioffe and Szegedy (2015), 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'

[77]: Srivastava et al. (2014), 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'

[78]: Murphy (2022), *Probabilistic Machine Learning: An introduction*

2.2 Robustness of Neural Networks

Seminal works [1, 50, 51] have shown that neural networks are vulnerable to small input perturbations, which can lead to misclassification.

Definition 2.2.1 (Adversarial Examples) *Given a norm $\|\cdot\|$ on the input space $\mathcal{X}$, a perturbation budget $\epsilon > 0$, a neural network f_θ we say that a perturbation $\delta \in \mathbb{R}^d$ is a successful* ***adversarial perturbation*** *of an input sample $(\boldsymbol{x}, y) \sim \mathcal{D}$ if:*

$$\|\delta\| \leq \epsilon \text{ and } \operatorname*{argmax}_{k \in [\![k]\!]} f_\theta(\boldsymbol{x} + \delta)[k] \neq y. \tag{2.5}$$

$\boldsymbol{x}_{adv} = \boldsymbol{x} + \delta$ is called an ***adversarial example*** *for the input $\boldsymbol{x}$ and label y.*

[1]: Goodfellow et al. (2015), 'Explaining and Harnessing Adversarial Examples'

[50]: Biggio et al. (2013), 'Evasion attacks against machine learning at test time'

[51]: Szegedy et al. (2014), 'Intriguing properties of neural networks'

Reminder: $B(\boldsymbol{x}, \epsilon)$ is defined as the ϵ-ball centered at $\boldsymbol{x}$ induced by the norm $\|\cdot\|$: $B(\boldsymbol{x}, \epsilon) := \{\boldsymbol{x}' \mid \|\boldsymbol{x} - \boldsymbol{x}'\| < \epsilon\}$.

Definition 2.2.2 (Local Robustness) *A neural network f_θ is* ***locally robust*** *at an input sample $(\boldsymbol{x}, y) \sim \mathcal{D}$, with respect to a perturbation budget $\epsilon > 0$, if no successful adversarial perturbation exists:*

$$\forall \boldsymbol{x}' \in B(\boldsymbol{x}, \epsilon) : \operatorname*{argmax}_{k \in [\![k]\!]} f_\theta(\boldsymbol{x}')[k] = y \tag{2.6}$$

Unless stated otherwise, we will consider the ℓ_∞ norm in the rest of this thesis. The problem of robust classification [37] is a modification of the standard supervised classification problem Eq. (2.3) where the goal is now to find parameters θ minimizing the loss computed over the worst case perturbation of the input samples:

$$\min_\theta \sum_{i=1}^{n} \max_{\boldsymbol{x}' \in B(\boldsymbol{x}^{(i)}, \epsilon)} \mathcal{L}\left(f_\theta(\boldsymbol{x}'), y^{(i)}\right). \tag{2.7}$$

The ℓ_∞ norm of a vector $\boldsymbol{x} \in \mathbb{R}^d$ is defined as $\|\boldsymbol{x}\|_\infty = \max_{i \in [\![d]\!]} |\boldsymbol{x}[i]|$.

[37]: Madry et al. (2018), 'Towards Deep Learning Models Resistant to Adversarial Attacks'

We refer to the loss function on the worst-case perturbation as the **robust loss**:

$$\mathcal{L}_{\text{rob}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) := \max_{\boldsymbol{x}' \in B(\boldsymbol{x}, \epsilon)} \mathcal{L}\left(f_\theta(\boldsymbol{x}'), y\right). \tag{2.8}$$

This min–max formulation requires solving a non-concave optimization problem in the inner maximization, which is generally intractable. Two popular families of methods address this challenge: under-approximating the inner maximization (**adversarial training**) or over-approximating it (**certified training**).

2.2.1 Adversarial Training

Adversarial training operates by first generating adversarial examples for each input sample, using an **adversarial attack algorithm** $\mathbb{A}$, and

then training the network on these adversarial examples instead of the original samples. The adversarial samples are generally recomputed at each training step to adapt to the current state of the network. A common approach to compute the attack is Projected Gradient Descent (PGD) [37], which iteratively perturbs the input sample with gradient steps, each followed by projection onto the ϵ-ball. Mathematically at step t the perturbation is computed as:

$$x^{t+1} = \Pi_{B(x,\epsilon)}\left(x^t + \alpha \operatorname{sign} \nabla_x \mathcal{L}\left(f_\theta(x^t), y\right)\right) \tag{2.9}$$

where $\Pi_{B(x,\epsilon)}$ is the projection onto the ϵ-ball centered at x, $\alpha > 0$ is the step size controlling the magnitude of the perturbation and sign is the sign function. At step $t = 0$, the perturbation is initialized with random uniform noise sampled from the ϵ-ball: $x^0 = x + \delta \sim \mathcal{U}(B(x, \epsilon))$. PGD can be implemented with multiple random restarts to increase the chances of finding a successful adversarial perturbation: the attack is run several times from different random initializations and the worst adversarial example is kept.

Given an adversarial attack oracle $\mathbb{A}$, we denote the adversarial sample of $(x, y) \sim \mathcal{D}$ as $x_{adv,\mathbb{A}}$ and define the corresponding adversarial loss as:

$$\mathcal{L}_{\text{adv}}(f_\theta, B(x, \epsilon); y) := \mathcal{L}(f_\theta(x_{adv,\mathbb{A}}), y). \tag{2.10}$$

For any adversarial attack oracle $\mathbb{A}$, we have $\mathcal{L}_{\text{adv}}(f_\theta, B(x, \epsilon); y) \leq \mathcal{L}_{\text{rob}}(f_\theta, B(x, \epsilon); y)$.

Such attacks can fool standardly trained neural networks and are widely used to train **empirically robust** networks. Empirically robust networks are typically evaluated on stronger attacks than those used during training. For instance, one might use PGD with many more steps and multiple restarts. AutoAttack [79] is a strong suite of attacks that has become the standard way to reliably evaluate empirical robustness.

Multi-step adversarial training incurs significant computational overhead, as training cost increases linearly with the number of attack steps.

2.2.2 Single Step Adversarial Training and Catastrophic Overfitting

A line of work has focused on designing effective single-step alternatives [4, 6, 61]. Among them is Fast Gradient Sign Method (FGSM) [1], which systematically lands on a corner of the perturbation space: $x_{\text{adv,FGSM}} = x + \epsilon \operatorname{sign}\left(\nabla_x \mathcal{L}(f_\theta(x), y)\right)$. FGSM was the first to be shown to suffer from catastrophic overfitting [6], a failure mode under which the network's robustness to PGD attacks rapidly drops to $\approx 0\%$. Understanding and mitigating this phenomenon is an activate area of research [83–86]. Wong et al. [6] empirically demonstrate that catastrophic overfitting can be prevented at moderate ϵ values by using a random uniform input in $B(x, \epsilon)$ as a FGSM starting point. This version of FGSM is known as *Random Start*-FGSM (RS-FGSM). Other techniques include explicit regularizers on the loss smoothness [87, 88], dynamically varying the attack step size [89], or selectively zeroing some coordinates of the attack step [90]. These are all outperformed by the noise-based N-FGSM: Jorge et al. [4] propose to sample uniformly from a larger set than the target model (typically, $B(x, 2\epsilon)$), and by removing RS-FGSM's projection onto $B(x, \epsilon)$ after the step. The resulting attack, called *Noisy*-FGSM (N-FGSM), achieves non-negligible PGD robustness at larger ϵ without incurring

[37]: Madry et al. (2018), 'Towards Deep Learning Models Resistant to Adversarial Attacks'

Projecting onto an ℓ_∞ ball simply consists in clipping each coordinate of the vector: $\Pi_{B(x,\epsilon)}(v = \min(\max(v, x - \epsilon), x + \epsilon)$.

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

[80]: Rice et al. (2020), 'Overfitting in adversarially robust deep learning'

[81]: Chen et al. (2020), 'Robust overfitting may be mitigated by properly learned smoothening'

[82]: Wang et al. (2024), 'Balance, imbalance, and rebalance: Understanding robust overfitting from a minimax game perspective'

Remark 2.2.1 Robust Overfitting It was noted that adversarial training is prone to overfitting with respect to empirical robustness, a phenomenon known as **robust overfitting** [80], with specialized remedies including smoothing methods [81] and techniques linked to a game-theoretical interpretation of the phenomenon [82]. Robust overfitting is a different phenomenon than catastrophic overfitting and can occur with multi-step adversarial training.

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

[61]: Shafahi et al. (2019), 'Adversarial training for free!'

[1]: Goodfellow et al. (2015), 'Explaining and Harnessing Adversarial Examples'

[83]: Vivek and Babu (2020), 'Single-step adversarial training with dropout scheduling'

[84]: Park and Lee (2021), 'Reliably fast adversarial training via latent adversarial perturbation'

[85]: Li et al. (2022), 'Subspace adversarial training'

[86]: Tsiligkaridis and Roberts (2022), 'Understanding and increasing efficiency of frank-wolfe adversarial training'

[87]: Sriramanan et al. (2020), 'Guided Adversarial Attack for Evaluating and Enhancing Adversarial Defenses'

[88]: Sriramanan et al. (2021), 'Towards Efficient and Effective Adversarial Training'

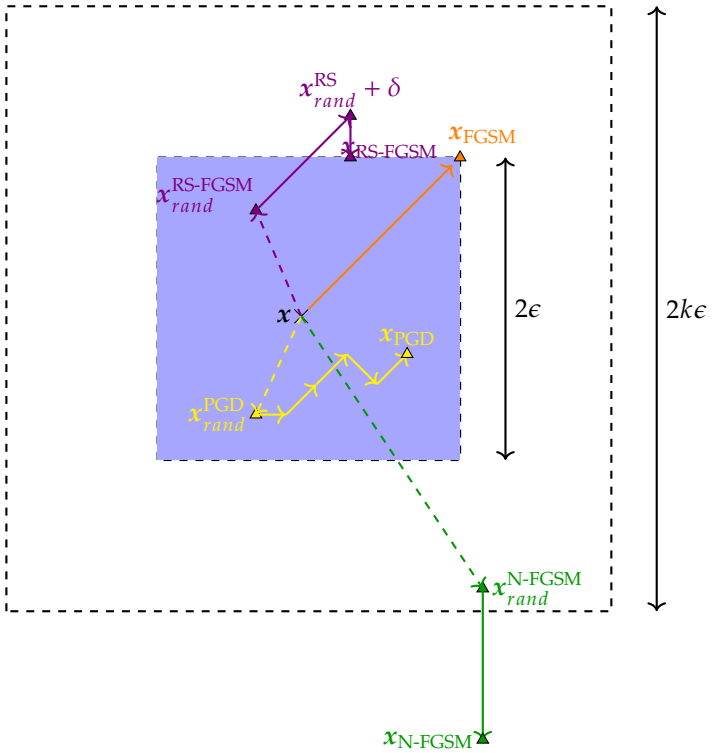


Figure 2.1: Illustration of the adversarial attacks discussed in this section. The blue square is the ball $B(\boldsymbol{x}, \epsilon)$ considered as the perturbation space. The initial random perturbation is indicated with a dashed-line. The gradient steps are indicated with arrows. See Algorithm 2.1 for the pseudocode of RS-FGSM. FGSM is obtained by removing the random initialization and setting $\alpha = \epsilon$. N-FGSM is obtained by initializing from a random point in $B(\boldsymbol{x}, k\epsilon)$ with $k > 1$ and omitting the projection step (line 8). PGD is obtained by iterating several times the steps at lines 6-8 with a smaller step size α.

additional cost relative to RS-FGSM, while also acting as an implicit loss regularizer [4].

See Figure 2.1 for an illustration of the single-step attacks and the PGD attack.

Regrettably, even N-FGSM suffers from catastrophic overfitting at larger perturbation radii.

Better robustness at large ϵ, albeit at additional cost, can be achieved by promoting the local linearity of the network, which is closely-linked with catastrophic overfitting [91]: GradAlign [5] aligns the input gradients of clean and randomly perturbed samples, ELLE [7] by enforcing a simple condition on two random samples from the perturbation set and their convex combination. Recent alternatives to attain large-ϵ robustness include hindering the generation of adversarial examples with a larger loss than the clean input [92], or the use of adaptive weight perturbations [93].

[89]: Kim et al. (2021), 'Understanding catastrophic overfitting in single-step adversarial training'

[90]: Golgooni et al. (2023), 'ZeroGrad: Costless conscious remedies for catastrophic overfitting in the FGSM adversarial training'

Listing 2.1: Pseudocode of the RS-FGSM attack.

```
1 function RS-FGSM(x, ϵ, α, f_θ)
2     Input: x, ϵ, α, f_θ
3     Output: x_adv
4
5     x_adv ← x + Uniform(−ϵ, ϵ)
6     g ← ∇_{x_adv} L(f_θ(x_adv), y)
7     x_adv ← x_adv + α · sign(g)
8     x_adv ←
        min( max(x_adv, x − ϵ), x + ϵ)
9
10    return x_adv
11 end function
```

2.2.3 Certified Training and Verified Robustness

On the other hand, works inspired by the formal verification literature have proposed to train neural networks by *over-approximating* the inner maximization problem in Eq. (2.7), thereby obtaining a lower bound on the loss that can be used to update the parameters θ.

Formal verification aims to provide guarantees of local robustness as defined in Eq. (2.6). Importantly, this notion is independent of any attack heuristic.

Definition 2.2.3 (Certified Robustness) *A network f_θ is said to be **certifiably robust** on an input sample $(\boldsymbol{x}, y) \sim \mathcal{D}$ if and only if the difference between the ground-truth logit $f_\theta(\boldsymbol{x}')[y]$ and every other logit is positive for*

[91]: Ortiz-Jimenez et al. (2023), 'Catastrophic overfitting can be induced with discriminative non-robust features'

[5]: Andriushchenko and Flammarion (2020), 'Understanding and Improving Fast Adversarial Training'

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'

[92]: Lin et al. (2023), 'Eliminating catastrophic overfitting via abnormal adversarial examples regularization'

[93]: Lin et al. (2024), 'Layer-Aware Analysis of Catastrophic Overfitting: Revealing the Pseudo-Robust Shortcut Dependency'

all $\boldsymbol{x}' \in B(\boldsymbol{x}, \epsilon)$*:*

$$\min_i \left\{ z_{f_\theta}^{B(\boldsymbol{x},\epsilon),y} := \min_{\boldsymbol{x}' \in B(\boldsymbol{x},\epsilon)} \left[z_{f_\theta}(\boldsymbol{x}', y) := \left(f_\theta(\boldsymbol{x}')[y]\mathbf{1} - f_\theta(\boldsymbol{x}') \right) \right] \right\} [i] \geq 0 \tag{2.11}$$

where $z_{f_\theta}(\boldsymbol{x}, y)$ *is the vector of differences between each logit and the ground-truth logit.*

This definition is equivalent to Eq. (2.6), but the formulation in terms of logit differences is more common in the verification literature. The reason for this will become clearer when we introduce verification methods. Given optimal bounds on the logit difference $z_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$, we can define a verified loss as:

$$\mathcal{L}_{\text{ver}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) := \mathcal{L}\left(-z_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}, y \right). \tag{2.12}$$

Example 2.2.1 (Logit Difference) For a neural network f_θ with 3 classes, consider the raw output logits $f_\theta(\boldsymbol{x}) = \begin{pmatrix} 2.0 \\ 1.5 \\ 0.5 \end{pmatrix}$ for an input $\boldsymbol{x}$ with ground-truth label $y = 0$. The logit difference is computed as:

$$z_{f_\theta}(\boldsymbol{x}, y) = \begin{pmatrix} 2.0 - 2.0 \\ 2.0 - 1.5 \\ 2.0 - 0.5 \end{pmatrix} = \begin{pmatrix} 0.0 \\ 0.5 \\ 1.5 \end{pmatrix}.$$

When $\mathcal{L}$ is a translation invariant loss function (which is the case for the cross-entropy loss), the verified loss of Eq. (2.12) is equal to the robust loss defined in Eq. (2.8):

$$\mathcal{L}_{\text{ver}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) = \mathcal{L}_{\text{rob}}(f_\theta, B(\boldsymbol{x}, \epsilon); y). \tag{2.13}$$

Of course computing the optimal adversarial perturbation and computing the exact bound of the logit difference is equivalently intractable. Certified training relaxes the problem by computing **sound** bounds of the logit difference over the perturbation space of Eq. (2.11): $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y} \leq z_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$. Given a bounding algorithm that produces such bounds, we can define an approximate verified loss as:

$$\mathcal{L}_{\underline{\text{ver}}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) := \mathcal{L}\left(-\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}, y \right) \tag{2.14}$$

Remark 2.2.2 In order to use the classical family of gradient descent methods to train a model with such a modified loss, the bounding algorithm must be differentiable.

For any adversarial attack oracle and any sound bounding algorithm, we have:

$$\mathcal{L}_{\text{adv}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) \leq \mathcal{L}_{\text{rob}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) \leq \mathcal{L}_{\underline{\text{ver}}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) \tag{2.15}$$

2.2.4 A first approximation: Interval Bound Propagation

[94]: Sunaga (1958), 'Theory of an interval algebra and its application to numerical analysis'

[95]: Moore (1966), *Interval Analysis*

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

Interval Bound Propagation (IBP) is a method to over-approximate neural network outputs based on the application of interval arithmetic [94, 95] onto the functions composing the neural architecture. As such, it can be easily applied onto general computational graphs [46]. For ease of presentation, let us assume that f_θ is a n-layer feed-forward neural network, with layer $j \in \{1 \dots n-1\}$ composed of an affine operation followed by a monotonically increasing element-wise activation function σ such as ReLU, $\boldsymbol{x}^j = \sigma\left(\boldsymbol{W}^i \boldsymbol{x}^{j-1} + \boldsymbol{b}^i\right)$, and a final affine layer: $f_\theta = \left(\boldsymbol{W}^n \boldsymbol{x}^{n-1} + \boldsymbol{b}^n\right)$. In this case, we can compose the logit differences and the last layer into a single affine layer: $z_{f_\theta}(\boldsymbol{x}, y) = \left(\tilde{\boldsymbol{W}}^n \boldsymbol{x}^{n-1} + \tilde{\boldsymbol{b}}^n\right)$.

In Example 2.2.1 we have

$$\tilde{\boldsymbol{W}}^n = \boldsymbol{W}^n \begin{pmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

and $\tilde{\boldsymbol{b}}^n = -\boldsymbol{b}^n$.

Then, for perturbation $B(\boldsymbol{x}^0, \epsilon)$, IBP computes $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ through the following procedure, which iteratively derives lower and upper bounds $\hat{\boldsymbol{l}}^k$ and $\hat{\boldsymbol{u}}^k$ to the outputs of the k-th affine layer:

$$\left.\begin{aligned}\hat{\boldsymbol{l}}^1 &= \boldsymbol{W}^1\boldsymbol{x}^0 - \epsilon\left|\boldsymbol{W}^1\right|\mathbf{1} + \boldsymbol{b}^1\\ \hat{\boldsymbol{u}}^1 &= \boldsymbol{W}^1\boldsymbol{x}^0 + \epsilon\left|\boldsymbol{W}^1\right|\mathbf{1} + \boldsymbol{b}^1\end{aligned}\right\} \quad \text{Initial bounds}$$

$$\left.\begin{aligned}\hat{\boldsymbol{l}}^j &= \tfrac{1}{2}\boldsymbol{W}^j\left(\sigma(\hat{\boldsymbol{l}}^{j-1}) + \sigma(\hat{\boldsymbol{u}}^{j-1})\right)\\ &\quad -\tfrac{1}{2}|\boldsymbol{W}^j|\left(\sigma(\hat{\boldsymbol{u}}^{j-1}) - \sigma(\hat{\boldsymbol{l}}^{j-1})\right) + \boldsymbol{b}_j\\ \hat{\boldsymbol{u}}^j &= \tfrac{1}{2}\boldsymbol{W}^j\left(\sigma(\hat{\boldsymbol{l}}^{j-1}) + \sigma(\hat{\boldsymbol{u}}^{j-1})\right)\\ &\quad +\tfrac{1}{2}|\boldsymbol{W}^j|\left(\sigma(\hat{\boldsymbol{u}}^{j-1}) - \sigma(\hat{\boldsymbol{l}}^{j-1})\right) + \boldsymbol{b}_j\end{aligned}\right\} \quad \forall\, j \in [\![2, n-1]\!]$$

$$\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y} = \frac{1}{2}\tilde{\boldsymbol{W}}^n\left(\hat{\boldsymbol{l}}^{n-1} + \hat{\boldsymbol{u}}^{n-1}\right) - \frac{1}{2}|\tilde{\boldsymbol{W}}^n|\left(\hat{\boldsymbol{u}}^{n-1} - \hat{\boldsymbol{l}}^{n-1}\right) + \tilde{\boldsymbol{b}}^n \quad \text{Final bounds} \tag{2.16}$$

We will henceforth write $\boldsymbol{l}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ for the lower bounds to the logit differences obtained through Eq. (2.16), and $\mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x},\epsilon); y) := \mathcal{L}\left(-\boldsymbol{l}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}, y\right)$ for the associated loss.

IBP is computationally inexpensive, requiring only one additional forward pass through the network, but suffers from bound explosion due to the **wrapping effect**[3]. Training using IBP requires a special initialization of the network weights [98] and a long schedule [99] with some epochs of standard training (*warm-up*) followed by epochs slowly increasing the perturbation budget ϵ (*ramp-up*) until final epochs of IBP training at the target perturbation budget ϵ.

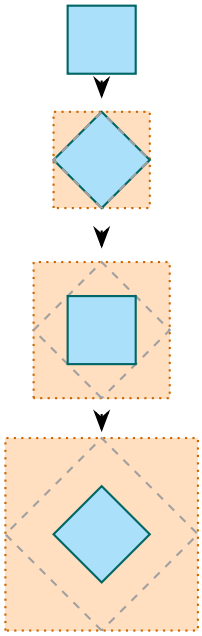


Figure 2.2: Illustration of the wrapping effect of interval arithmetic. The initial space is a product of two intervals (cyan square). The cyan squares across panels are the exact reachable sets after repeated 45-degree rotations. The orange squares (enclosing boxes) are the over-approximated reachable sets obtained with interval arithmetic. The gray dashed lines are the exact reachable sets obtained by applying the rotation on the previous enclosing box. The wrapping effect (the over-approximation of the box enclosure) leads to a rapid explosion of the approximated reachable set. This is highly related to the **dependency problem**, a more general pitfall of interval analysis. Consider a variable x taking possible values in an interval $[0, 1]$. Interval arithmetic gives $x - x \in [-1, 1]$, whereas an abstraction that tracks variable relations yields the exact result $[0, 0]$.

2.3 Hybrid Methods: state of the art in certified training

Networks trained via the IBP loss $\mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x},\epsilon); y)$ can be easily verified to be robust using the same bounds employed for training. However, the relative looseness of the over-approximation from equation Eq. (2.16) results in a significant decrease in standard performance. For ReLU networks, the availability of more effective verifiers based on branch-and-bound [41, 100, 101] post-training has motivated the development of alternative techniques (see Section 3.5 for more details on branch-and-bound approaches). A series of methods have shown that combination of adversarial training with network over-approximations [97, 102–104] can yield strong verifiability via branch-and-bound while reducing the impact on standard performance.

2.3.1 SABR: Small Adversarial Bounding Regions

In particular, Müller et al. [105] proposed to compute the IBP loss over a tunable subset of $B(\boldsymbol{x},\epsilon)$ containing adversarial examples, yielding favorable trade-offs between standard performance and certified robustness via branch-and-bound-based methods. The method is dubbed **SABR**

3: The term comes from the field of reachability analysis in dynamic systems [96]. It is particularly striking for interval analysis but any domain introducing over-approximations, *wrapping* more than necessary, can suffer from it.

[97]: Fan and Li (2021), 'Adversarial Training and Provable Robustness: A Tale of Two Objectives'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

(Small Adversarial Bounding Regions). The subset used as initial bounds for IBP is a ℓ_∞-ball of center $\boldsymbol{x}_{\text{adv}}$, an adversarial example typically computed by PGD, and of radius $\lambda\epsilon$ with $\lambda \in [0, 1]$ a tunable parameter, as schematized in Fig. 2.3.

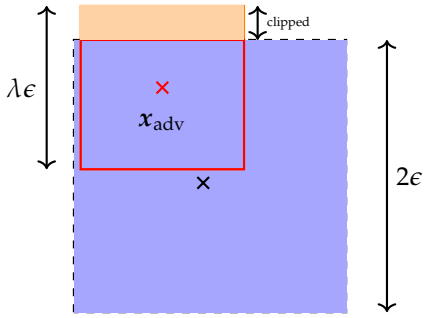


Figure 2.3: SABR: the blue box represent the perturbation space $B(\boldsymbol{x}, \epsilon)$ of the input sample $\boldsymbol{x}$. The red cross $\boldsymbol{x}_{\text{adv}}$ is an adversarial example, considered as center for a new, smaller box to be used for training, delimited by the red lines. This box is clipped to the original perturbation space.

2.3.2 Expressive Losses

De Palma et al. [8, Definition 3.1] introduced the notion of **loss expressivity**, defined as the ability of a loss function to spawn a continuous range of trade-offs between the adversarial and the IBP losses, showing that expressive losses obtained through convex combinations between an adversarial and an over-approximation component lead to state-of-the-art certified robustness.

Definition 2.3.1 (Expressive losses) *Given a bounding algorithm and an adversarial attack generator a parametrized family of loss functions $\mathcal{L}_\alpha$ is said to be **expressive** if:*

- $\mathcal{L}_{adv}(f_\theta, B(\boldsymbol{x}, \epsilon); y) \leq \mathcal{L}_\alpha(f_\theta(\boldsymbol{x}); y) \leq \mathcal{L}_{\underline{ver}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)$ *for all* $\alpha \in [0, 1]$.
- $\mathcal{L}_\alpha(f_\theta(\boldsymbol{x}); y)$ *is continuous and monotonically increasing with respect to* $\alpha \in [0, 1]$.
- $\mathcal{L}_0(f_\theta(\boldsymbol{x}); y) = \mathcal{L}_{adv}(f_\theta, B(\boldsymbol{x}, \epsilon); y)$
- $\mathcal{L}_1(f_\theta(\boldsymbol{x}); y) = \mathcal{L}_{\underline{ver}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)$.

De Palma et al. [8] introduce several ways to combine the adversarial loss and the IBP loss to obtain expressive losses using convex combinations.

Example 2.3.1 (Convex combinations) The Multi-Task Learning IBP (MTL-IBP) loss is defined as the convex combination of the adversarial loss and the IBP loss:

$$\mathcal{L}_{\alpha,MTL-IBP} := (1 - \alpha) \cdot \mathcal{L}_{\text{adv}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) + \alpha \cdot \mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) \tag{2.17}$$

The convex combination can also be made in the logarithmic space:

$$\mathcal{L}_{\alpha,EXPIBP} := \mathcal{L}_{\text{adv}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)^{1-\alpha} \cdot \mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)^{\alpha}. \tag{2.18}$$

The combination can also be done between the logit differences on the adversarial input on one hand, and the lower bound of the logit differences on the clean input on the other hand:

$$\mathcal{L}_{\alpha,CC-IBP} := \mathcal{L}\left(-\left[(1 - \alpha) \cdot \boldsymbol{z}_{f_\theta}(\boldsymbol{x}_{\text{adv}}, y) + \alpha \cdot \underline{\mathbf{z}}_{f_\theta}(\boldsymbol{x}, y)\right]; y\right) \tag{2.19}$$

The formulation of MTL-IBP stems from multi-task learning [106]. Here the two tasks are the minimization of the adversarial loss and the IBP loss. CC-IBP on the other hand is inspired by a previous work [107] on efficient and precise certified training, which does a convex combination of the IBP bounds and CROWN-IBP bounds inside the loss (we describe CROWN-IBP in Section 2.4.2). Finally, Exp-IBP was first introduced to solidify the claim that expressiveness is a key property of losses for certified training [8]. Its reduced sensitivity to the changes of α can be interesting when tuning it as we will see in Chapter 4.

The hybrid use of adversarial training and certified training has been shown to yield state-of-the-art results when evaluated with verification

[41]: Ferrari et al. (2022), 'Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound'

[100]: Henriksen and Lomuscio (2021), 'DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis'

[101]: Wang et al. (2021), 'Improving Global Adversarial Robustness Generalization With Adversarially Trained GAN'

[97]: Fan and Li (2021), 'Adversarial Training and Provable Robustness: A Tale of Two Objectives'

[102]: Balunovic and Vechev (2020), 'Adversarial Training and Provable Defenses: Bridging the Gap'

[103]: De Palma et al. (2022), 'IBP Regularization for Verified Adversarial Robustness via Branch-and-Bound'

[104]: Mao et al. (2023), 'TAPS: Connecting Certified and Adversarial Training'

[105]: Müller et al. (2023), 'Certified Training: Small Boxes are All You Need'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[106]: Caruana (1997), 'Multitask Learning'

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

Table 2.1: Comparison of the expressive losses from with literature results for ℓ_∞ norm perturbations on CIFAR-10, TinyImageNet and downscaled (64×64) ImageNet. The entries corresponding to the best standard or verified robust accuracy for each perturbation radius are highlighted in bold.

Dataset	ϵ	Method	Source	Standard acc. [%]	Verified rob. acc. [%]
CIFAR-10	$\frac{2}{255}$	CC-IBP	De Palma et al. [8]	80.09	**63.78**
		MTL-IBP	De Palma et al. [8]	80.11	63.24
		Exp-IBP	De Palma et al. [8]	**80.61**	61.65
		STAPS	Mao et al. [104]	79.76	62.98
		SABR	Mao et al. [109]	79.89	63.28
		SortNet	Zhang et al. [108]	67.72	56.94
		IBP-R	Mao et al. [109]	80.46	62.03
		IBP	Mao et al. [109]	68.06	56.18
		AdvIBP	Fan and Li [97]	59.39	48.34
		CROWN-IBP	Zhang et al. [107]	71.52	53.97
		COLT	Balunovic and Vechev [102]	78.4	60.5
	$\frac{8}{255}$	CC-IBP	De Palma et al. [8]	53.71	35.27
		MTL-IBP	De Palma et al. [8]	53.35	35.44
		Exp-IBP	De Palma et al. [8]	53.97	35.04
		STAPS	Mao et al. [104]	52.82	34.65
		SABR	Müller et al. [105]	52.38	35.13
		SortNet	Zhang et al. [108]	**54.84**	**40.39**
		IBP-R	De Palma et al. [103]	52.74	27.55
		IBP	Shi et al. [98]	48.94	34.97
		AdvIBP	Fan and Li [97]	47.14	33.43
		CROWN-IBP	Xu et al. [46]	46.29	33.38
		COLT	Balunovic and Vechev [102]	51.70	27.50
TinyImageNet	$\frac{1}{255}$	CC-IBP	De Palma et al. [8]	38.61	**26.39**
		MTL-IBP	De Palma et al. [8]	37.56	26.09
		Exp-IBP	De Palma et al. [8]	38.71	26.18
		STAPS	Mao et al. [104]	28.98	22.16
		SABR	Mao et al. [109]	28.97	21.36
		SortNet	Zhang et al. [108]	25.69	18.18
		IBP	Mao et al. [109]	25.40	19.92
		CROWN-IBP	Shi et al. [98]	25.62	17.93
ImageNet64	$\frac{1}{255}$	CC-IBP	De Palma et al. [8]	19.62	11.87
		MTL-IBP	De Palma et al. [8]	20.15	12.13
		Exp-IBP	De Palma et al. [8]	**22.73**	**13.30**
		SortNet	Zhang et al. [108]	14.79	9.54
		CROWN-IBP	Xu et al. [46]	16.23	8.73
		IBP	Gowal et al. [99]	15.96	6.13

methods. We report the results on several image classification datasets as shown in De Palma et al. [8] in Table 2.1.

For more details on the methods and the results of Table 2.1 we refer to De Palma et al. [8]. Briefly: AdvIBP [97], CROWN-IBP [107], COLT [102], IBP-R [103], STAPS [104] all relies in some way on approximating the network output bounds (using IBP or tighter convex relaxation), while SortNet [108] is a special architecture: a 1-Lipschitz neural network, providing guarantees of robustness thanks to the low Lipschitz constant of the network.

[97]: Fan and Li (2021), 'Adversarial Training and Provable Robustness: A Tale of Two Objectives'

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

[102]: Balunovic and Vechev (2020), 'Adversarial Training and Provable Defenses: Bridging the Gap'

[103]: De Palma et al. (2022), 'IBP Regularization for Verified Adversarial Robustness via Branch-and-Bound'

[104]: Mao et al. (2023), 'TAPS: Connecting Certified and Adversarial Training'

2.4 Bound Propagation: over-approximating neural networks

As described in Section 2.2.4 a first way to compute an over-approximation of neural networks bounds is to use interval arithmetics to propagate bounds in the network, commonly referred to as **Interval Bound Propagation** (IBP). While the method remains popular for training networks and verifying networks trained with IBP, the imprecision of the bounds when applied to networks trained in a standard way makes it impractical for verification. We describe the general framework of linear approximation algorithms offering tighter bounds at a higher computational cost. They all share the idea of keeping track of the relations between the variables (the inputs and the intermediate nodes) of the networks.

2.4.1 Neural Networks as computational graphs

While we have until now considered neural networks as sequence of matrix multiplication and applications of activation functions, we can also represent them as graphs. We represent such a network in Figure 2.4.

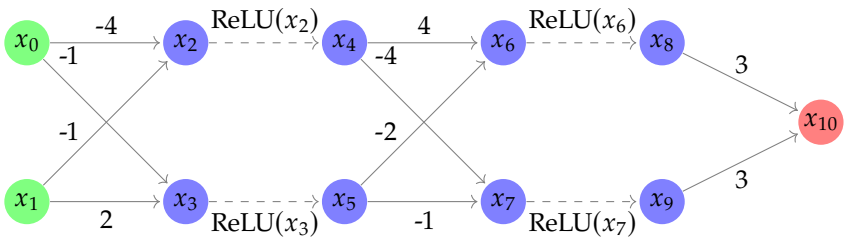


Figure 2.4: A simple neural network with 2 hidden layers, input of dimension 2 and a single output. Following the notations of Example 2.1.1, the weight matrices are: $\left\{\begin{pmatrix}-4 & -1\\ -1 & 2\end{pmatrix}, \begin{pmatrix}4 & -2\\ -4 & -1\end{pmatrix}, \begin{pmatrix}3 & 3\end{pmatrix}\right\}$, the biases are omitted for clarity. The input is represented in green, the hidden layers in blue and the output in red.We decouple the application of the linear function from the activation function, deviating from the common graph representation to better illustrate how non-linearities are handled when approximating the reachable space of a network. The nodes immediately before the activation functions are called Pre-Activation nodes (or Pre-ReLU in the case of ReLU activations).

In the following we consider a neural network f_θ with a total of N nodes, and L layers as defined in Eq. (2.1). We consider a simple sequential neural network that can be decomposed as:

$$f_\theta(\boldsymbol{x}) = \left(f_L \circ f_{L-1} \circ \cdots \circ f_2 \circ f_1\right)(\boldsymbol{x}).$$

4: The vectors $\boldsymbol{x}^k$ are typically called **intermediate** or **latent** features for hidden layers $k < L$.

for a layer $k \in [\![1]\!]$ we note ${f_\theta}^k$ the function that computes the output values of the layer k when applied to the inputs and denote its output $\boldsymbol{x}^k$[4]:

$$\boldsymbol{x}^k = {f_\theta}^k(\boldsymbol{x}) = \left(f_k \circ f_{k-1} \cdots \circ f_1\right)(\boldsymbol{x}).$$

We note $\mathcal{I}_k$ the set of indices of the nodes of f_θ at layer k, with $\mathcal{I}_0 = \{0, \ldots, n_0 - 1\}$ for a network with input dimension n_0.

2.4.2 Linear Approximations for Neural Network Verification

The over-approximation of the reachable space is one of the most effective ways to overcome the scalability problem of verifying neural networks. The goal is to compute a *conservative* over-approximation of the reachable space, such that any concrete behavior is necessarily included in the approximation.

More formally, we note $f_\theta(\mathcal{X}) := \left\{f_\theta(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}\right\}$ the exact reachable space of a neural network f_θ, and $\mathcal{S}$ a safe subset of the output space of f_θ[5]. Approximating the reachable space is done by computing a set $\mathcal{O}_k^\sharp$ that over-approximates the reachable set of the network at layer k for every layer of the network: ${f_\theta}^k(\mathcal{X}) \subset \mathcal{O}_k^\sharp$. At the output layer we then have ${f_\theta}^L(\mathcal{X}) = f_\theta(\mathcal{X}) \subset \mathcal{O}_L^\sharp$. Checking that the network is safe can then be done by checking that $\mathcal{O}_L^\sharp \subset \mathcal{S}$. When the approximation is too loose (where both $\mathcal{S} \cap \mathcal{O}_L^\sharp$ and $\mathcal{S}^c \cap \mathcal{O}_L^\sharp$) it may be impossible to conclude, hence the importance of computing tight approximations.

5: In the case of local robustness, $\mathcal{X}$ is an epsilon ball $B(\boldsymbol{x}, \epsilon)$ and $\mathcal{S}$ is the set of logit vectors such that $\boldsymbol{z}_{f_\theta}(\boldsymbol{x}, y) \geq 0$ for the target class y.

We have in fact already described such an over-approximation in Eq. (2.16), where the reachable space is approximated by a box $\mathcal{O}_k^\sharp = \prod_{i \in \mathcal{I}_k} [l_i, u_i]$ for each layer k in the network using interval arithmetics. However, the computed bounds are typically too loose to prove properties of interest, particularly for networks not trained with certified training.

We now describe the general framework of linear approximations to compute tighter bounds by retaining (possibly approximated) linear relations between the nodes of the network.

The key idea is to approximate the behavior of the activation functions with a linear approximation. Starting with the inputs, concrete bounds and symbolic relations are computed and propagated through the layers of the network to obtain bounds on each output, which can be used to check whether the property of interest holds.

We explain this approach following the bound propagation perspective of Singh et al. [40]. Let x_i be a node in the network at a layer k ($i \in \mathcal{I}_k$), the goal is to compute and propagate the following:

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

$$
\begin{aligned}
x_i &\leq b^u + \sum_{j<i} \lambda_j^u x_j \\
x_i &\geq b^l + \sum_{j<i} \lambda_j^l x_j \\
l_i &\leq x_i \leq u_i
\end{aligned}
$$

where $l_i, u_i \in \mathbb{R}$ are concrete bounds on x_i, $\lambda_j^u, b^u \in \mathbb{R}$ are coefficients representing the upper symbolic relations of x_i with regards to previous nodes, and $\lambda_j^l, b^l \in \mathbb{R}$ the lower symbolic relations. For simplicity, we denote the upper symbolic relations of a node x_i as $\overline{x_i} = b^u + \sum_{j<i} \lambda_j^u x_j$ and the lower symbolic relations as $\underline{x_i} = b^l + \sum_{j<i} \lambda_j^l x_j$.

Remark 2.4.1 (Soundness) Let x_j, $j \in [\![n]\!]$ be a node in the network, and l_j, u_j its concrete bounds. Let $k \in [\![l]\!]$ be the layer where x_j is in the network, n_k be the output dimension of ${f_\theta}^k$ and $m \in [\![n_k]\!]$ be the index of the output node x_j. Let $\boldsymbol{x}$ be an input sample and $\mathcal{I}$ be the input space of the network defined by a safety property (*e.g.* an ℓ_∞ ball around $\boldsymbol{x}$). We say that an analysis maintaining such bounds and symbolic relations is **sound** if for any node x_j in the network, the following holds:

$$\forall \boldsymbol{x}' \in \mathcal{I}, l_j \leq {f_\theta}^k(\boldsymbol{x}')[m] \leq u_j. \tag{2.20}$$

Propagating the bounds and symbolic relations

At the first hidden layer $k = 1$ the lower and upper symbolic relations are initialized with the layer weights and biases while the concrete bounds are initialized multiplying the input bounds with the weights and adding the biases following classical interval arithmetics.

For subsequent linear layers, propagating the symbolic relations is straightforward. The new upper symbolic relations are computed by multiplying the previous upper symbolic relations with the positive part of the weights and the previous lower symbolic relations with the negative part of the weights. The new lower symbolic relations are computed analogously.

For non-linear functions different techniques can be employed. We highlight the specific treatment of the ReLU activation function.

Definition 2.4.1 *Let x_i be a pre-ReLU node with bounds l_i, u_i. The ReLU activation can be categorized in three cases:*

1. ***Always deactivated**: if $u_i \leq 0$, then the ReLU is always deactivated.*
2. ***Always activated**: if $l_i > 0$, then the ReLU is always activated.*
3. ***Uncertain** or **unstable**: if $l_i < 0 < u_j$ there is no guarantee that the ReLU is always activated or always deactivated.*

When the ReLU activation status is stable the propagation is straightforward:

- If the ReLU is always deactivated, then the output is a constant zero function, symbolic relations are dropped and both concrete bounds are set to zero.
- If the ReLU is always activated, then it is treated as the identity function, the symbolic relations are kept and the concrete bounds are set to the pre-ReLU bounds.

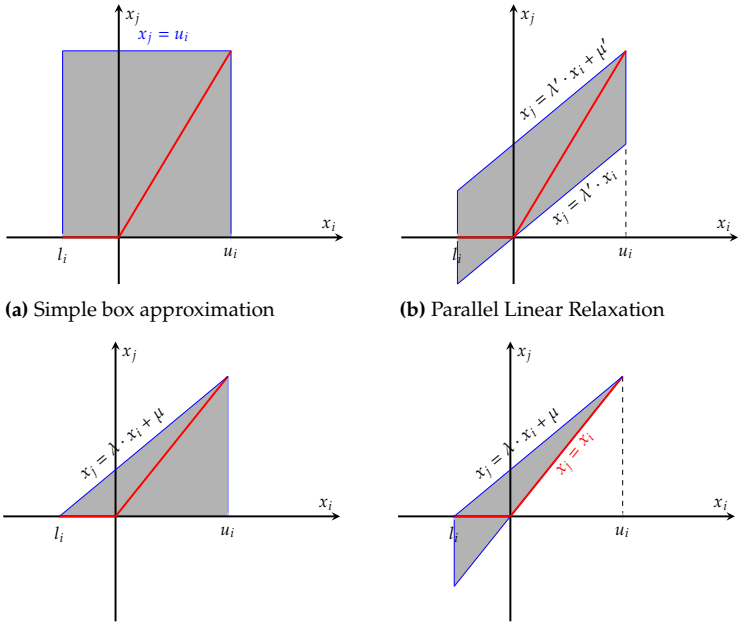
In the uncertain case different linear approximations can be used.

We illustrate the main ones in the single-variable case in Figure 2.5. In Figure 2.5a the relations are dropped, and a new fresh symbol is created and used in subsequent operations. In this case, relations with regard to inputs can be lost, but introducing fresh symbols allows for some tightening of the bounds in subsequent operations. This is the technique at the heart of the tool ReluVal [58] and further described in [110]. It is commonly referred to as **Symbolic Interval Propagation**.

In Figure 2.5b a **parallel linear approximation** is used. A slope λ is introduced and used in both the update of the lower and upper symbolic

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

[110]: Li et al. (2019), 'Analyzing Deep Neural Networks with Symbolic Propagation: Towards Higher Precision and Faster Verification'



(a) Simple box approximation

(b) Parallel Linear Relaxation

(c) Approximation if $u_i \leq -l_i$

(d) Approximation if $u_i > -l_i$

Figure 2.5: Linear approximations of the ReLU function

coefficients. We show here the approximation with $\lambda = \frac{u_i}{u_i - l_i}$, which is the common choice in the literature [39, 42, 43, 111]. But any λ such that $0 \leq \lambda \leq 1$ can be used and will yield a sound approximation.

Finally, one can also use a different slope for the lower and upper symbolic relations, as done in Singh et al. [40] (the method is called **DeepPoly**) and Zhang et al. [45] (the method is called **CROWN**). The upper slope is typically chosen as $\lambda^u = \frac{u_i}{u_i - l_i}$, as in the parallel linear approximation, while the lower slope is chosen heuristically to minimize the area of the over-approximation (the gray area in Figure 2.5), with the choices of $\lambda^l = 0$ if $u_i \leq -l_i$ (Figure 2.5c) and $\lambda^l = 1$ if $l_i \geq 0$ (Figure 2.5d).

Here again any choice of slope λ^l such that $0 \leq \lambda^l \leq 1$ is sound. While there is no theoretical guarantee providing the slope leading to the tightest output bounds, the heuristics of Singh et al. [112] and Zhang et al. [45], minimizing the area of the over-approximation, have been shown to yield good bounds in practice. Optimal slopes can also be learned, as discussed in paragraph Optimizing the slopes.

Backsubstitution The symbolic relations can be maintained as to relate directly to the inputs at every step of the bound propagation procedure, using the input relations of the previous layer. However, this prevents further tightening of the bounds due to possible intermediate simplifications. To obtain the tightest possible bounds, a procedure that recursively substitutes relations between consecutive layers, *going through all previous layers for each intermediate layer*, is used in Singh et al. [40], Wong and Kolter [42], Weng et al. [43], and Zhang et al. [45]. Singh et al. [40] calls it **backsubstitution** and Zhang et al. [45] calls it **backward mode**. The forward mode described in Zhang et al. [45] amounts to directly updating relations to inputs in a single forward pass using the rules described above. While the backsubstitution procedure yields better bounds it requires a quadratic number of backward passes. Zhang et al. [107] propose a tradeoff with **CROWN-IBP**: computing intermediate

[39]: Singh et al. (2018), 'Fast and Effective Robustness Certification'

[42]: Wong and Kolter (2018), 'Provable defenses against adversarial examples via the convex outer adversarial polytope'

[43]: Weng et al. (2018), 'Towards fast computation of certified robustness for ReLU networks'

[111]: Wang et al. (2018), 'Efficient Formal Safety Analysis of Neural Networks'

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

[45]: Zhang et al. (2018), 'Efficient Neural Network Robustness Certification with General Activation Functions'

[112]: Singh et al. (2019), 'Boosting Robustness Certification of Neural Networks'

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

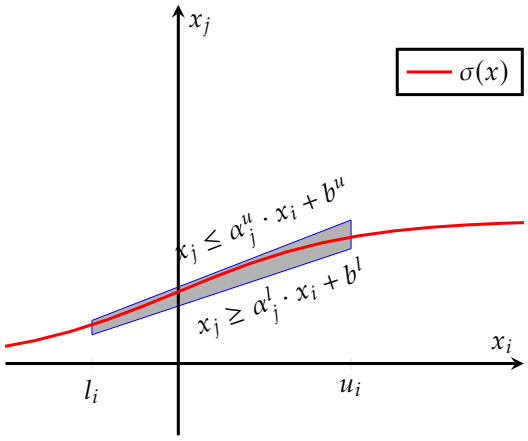


Figure 2.6: Linear approximation of the Sigmoid function

bounds with IBP and using those bounds for relaxing the non-linearities when backsubstituting from the last layer to the first layer.

Note that Singh et al. [39], improving on Gehr et al. [38], uses a zonotope representation [113, 114]. This allows to compute the same reachable space as the parallel linear approximation, without the need to backsubstitute, but requires introducing and tracking relations with new symbols for every unstable ReLU nodes.

When the output set of interest involves affine comparisons between outputs of the network a final refinement can be made. The specification of the output space of interest can be treated as an additional final layer and the bounds and symbolic relations can be further propagated through this layer[6].

Many of the linearization techniques were introduced in concurrent works, inspired by different views (notably abstract interpretation [39, 40] and optimization [42, 45]), leading to somewhat different presentations and implementations of similar ideas. We refer to Salman et al. [115] for a detailed discussion of the different linearization techniques and how they relate to each other. Li et al. [116] systematically evaluates several linearization techniques (along with other methods) and experimentally shows the equivalence of DeepPoly [40] and CROWN [45] or Fast-Lin [43] and WK [42][7].

General activation functions We focused on the ReLU activation function as it is widely used in practice and most early works focused on it. Other piecewise linear activations, can be handled similarly, considering the different cases where the linear relations can be propagated exactly, or when a new approximation is needed. Such activations include Leaky ReLU [117], PReLU [118], or ReLU6 [119][8]. Other activation functions such as Sigmoid or Tanh can also be linearly approximated with either parallel or non-parallel linear approximations as shown in Figure 2.6. Many different heuristics can be used to choose the slopes for those approximations [39, 45, 56, 120–123].

2.4.3 Linear approximations and certified training

Note that linear approximations techniques are in general differentiable. Implemented with a framework supporting automatic differentiation (*e.g.* Auto_LiRPA [46], built with PyTorch [124]) the methods can be used to train networks with verifiability goals as described in Section 2.2.3.

[39]: Singh et al. (2018), 'Fast and Effective Robustness Certification'

[38]: Gehr et al. (2018), 'AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation'

[113]: Girard (2005), 'Reachability of Uncertain Linear Systems Using Zonotopes'

[114]: Ghorbal et al. (2009), 'The Zonotope Abstract Domain Taylor1+'

6: In the case of local robustness this amounts to using the logit difference defined in Eq. (2.11). One can use a final linear layer with weights $W \in \mathbb{R}^{K \times K}$ for K classes where the column corresponding of the target class is filled with 1 and the diagonal is filled with -1 except for the row corresponding to the target class which is filled with 0, as in Eq. (2.2.4).

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

[42]: Wong and Kolter (2018), 'Provable defenses against adversarial examples via the convex outer adversarial polytope'

[45]: Zhang et al. (2018), 'Efficient Neural Network Robustness Certification with General Activation Functions'

[115]: Salman et al. (2019), 'A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks'

[116]: Li et al. (2020), 'SoK: Certified Robustness for Deep Neural Networks'

[43]: Weng et al. (2018), 'Towards fast computation of certified robustness for ReLU networks'

7: The results are available at `https://sokcertifiedrobustness.github.io/benchmark/`. Some theoretically equivalent methods yield different results when timeout is a factor, but same results on the easiest instances.

[117]: Maas (2013), 'Rectifier Nonlinearities Improve Neural Network Acoustic Models'

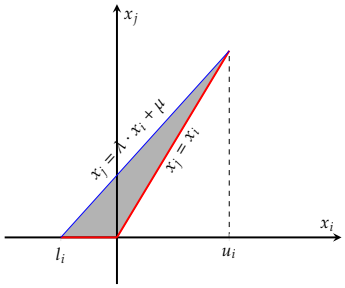


Figure 2.7: Optimal convex approximation

In fact, it is the purpose of the first introduction of the parallel linear approximation in Wong and Kolter [42].

However, tighter bounds do not necessarily lead to better certified model. IBP, especially combined with adversarial training, remains the leading bound propagation method used for training as shown in Table 2.1. Jovanović et al. [125] argues that two properties, met by IBP, **continuity** and **sensitivity**, are keys for well-behaved certified training dynamics. They show that the linear approximations methods do not satisfy either one or both of these properties, thus leading to problems in the training process.

2.4.4 Beyond linear approximations

We now briefly describe some variations of the linear approximations described above.

Tighter approximations It is possible to use tighter approximations by using more than one upper or lower symbolic relations per node. For instance for the ReLU functions the optimal relaxation (minimizing the area of the over-approximation) is the triangle relaxation [126], using two lower symbolic relations representing the convex hull of the graph of the ReLU function as shown in Figure 2.7. The relaxation is typically used to formulate the bounding problem as a Linear Program (LP) or a Mixed Integer Linear Program (MILP). We defer the discussion of such methods in Section 3.5.

Another approach is to use quadratic [45] or polynomial approximations [127]. However, computing bounds using such approximation is typically quite expensive, and the methods are limited to small networks, particularly in the context of networks used within a cyber-physical system where polynomial relations are used to model the dynamics of the system.

Combining linear approximations There is no theoretical guarantees showing an approximation to produce always tighter bounds than another. In practice even for the same network one method might yield better bounds for a node than another. Mazzucato and Urban [57] propose to use jointly different linear approximations, referred to as **abstract domains**, taking the best bounds at every step of the analysis[9].

[118]: He et al. (2015), 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification'

[119]: Krizhevsky (2010), 'Convolutional deep belief networks on cifar-10'

8: All are variants of ReLU. Leaky ReLU uses a small positive slope for negative inputs. Parametric ReLU (PReLU) uses the same idea but makes it a tunable parameter. ReLU6 clips positive inputs to 6.

[56]: Wu et al. (2022), 'Toward Certified Robustness Against Real-World Distribution Shifts'

[120]: Henriksen and Lomuscio (2020), 'Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search'

[121]: Zhang et al. (2022), 'Provably Tightest Linear Approximation for Robustness Verification of Sigmoid-like Neural Networks'

[122]: Zhang et al. (2024), 'GaLileo: General Linear Relaxation Framework for Tightening Robustness Certification of Transformers'

[123]: Shi et al. (2025), 'Neural Network Verification with Branch-and-Bound for General Nonlinearities'

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

[124]: Paszke et al. (2019), 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'

[125]: Jovanović et al. (2022), 'On the Paradox of Certified Training'

[126]: Elboher et al. (2019), 'An Abstraction-Based Framework for Neural Network Verification'

[127]: Kochdumper et al. (2023), 'Open- and closed-loop neural network verification using polynomial zonotopes'

[57]: Mazzucato and Urban (2021), 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'

Optimizing the slopes Rather than choosing fixed slopes heuristically Xu et al. [128] propose to optimize them in a method called α-**CROWN**[10]. The differentiable nature of the linear approximations makes it possible to use gradient methods to find approximate solutions within a given number of iterations, maintaining sound bounds at every step. The additional cost scales linearly with the number of iterations. While there is no convergence guarantees to the optimal slopes, Xu et al. [128] show experimentally that the method improves bounds significantly, sometimes yielding better bounds than the ones obtained by solving the LP problem of the triangle relaxation.

9: The method is used to verify fairness properties of neural networks and not safety properties. We briefly describe the approach in Section 3.4.4.

[128]: Xu et al. (2021), 'Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers'

10: The α denotes the slopes in their paper.

Tightening Bounds for Incomplete Verification

An input partitioning heuristic for the verification of neural networks

3

In this chapter we present a heuristic to accelerate the partitioning of the input space of a neural network, allowing efficient verification of neural networks with low-dimensional inputs. The heuristic is coined ReCIPH (*Relational Coefficient for an Input Partitioning Heuristic*), implemented in the Python Reachability Assessment Tool (PyRAT) [59]. We show that the heuristic outperforms the natural baseline of choosing the widest interval to split on, and the gradient smear heuristic [58]. We also show the use of ReCIPH beyond safety properties: it also helps in the context of fairness verification, where the input partitioning is used to certify fairness properties of neural networks [57]. This work [129] was presented as a poster at the First Workshop on Formal Verification and Machine Learning (WFVML 2022), organized in conjunction with ICML 2022.

3.1 Context and Motivation

We focus in this work on refining the bounds obtained with linearization techniques described in Section 2.4.2.

Bound propagation and linearization techniques are powerful particularly to analyze large neural networks, with high-dimensional inputs. In some contexts, with networks trained with verifiability goals, verification with the bounds obtained using only such techniques yields verification results close to the results obtained with state-of-the-art branch and bound techniques[1].

However, bound propagation trades precision for scalability. This is particularly problematic for networks with no robustness induced during training. The problem is exacerbated when the perturbed input space is large. In this case, even for small networks, the bounds obtained with linearization techniques can be too loose to prove interesting properties. For instance, the staple benchmark ACAS Xu, introduced in [130], is a set of 45 networks each with 5 inputs and 6 hidden layers of 50 nodes. While the networks are small, analysis using only linearization techniques fails to prove the properties introduced in [131]. Several use cases in the latest edition of International Verification of Neural Networks Competition (VNN-COMP 2024) [60] comprise neural networks with less than 30 inputs.

3.1.1 Input Partitioning

For networks with low-dimensional inputs, a simple but powerful technique to overcome the limited precision of linearization techniques is input space partitioning [58]. The subspaces of the partition are treated as input spaces for new verification problems. This can be naturally implemented in a parallelized manner. If the property holds on every subspace of the partition, then the property holds for the original input space. It has been shown in Wang et al. [58] that input partitioning combined with symbolic interval propagation enables asymptotically complete analysis: it is possible to compute an arbitrarily precise over-approximation of the

[59]: Lemesle et al. (2024), 'Neural Network Verification with PyRAT'

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

[57]: Mazzucato and Urban (2021), 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'

[129]: Durand et al. (2022), 'ReCIPH: Relational Coefficients for Input Partitioning Heuristic'

1: We refer for instance to the table 6 in appendix G.3 of De Palma et al. [8] for a comparison of branch and bound, CROWN with IBP used for intermediate bounds and IBP only for assessing the verified accuracy of models trained with expressive losses. On CIFAR-10 with a target $\epsilon = 8/255$ IBP and CROWN-IBP yield identical results, and BaB only improves about 1-2 percentage point (from 34% to 35% verified accuracy). On TinyImageNet with a target $\epsilon = 1/255$, CROWN-IBP (23%) greatly improves the results of IBP (< 1%) and is about 3 percentage points behind BaB (26%).

[130]: Julian et al. (2016), 'Policy compression for aircraft collision avoidance systems'

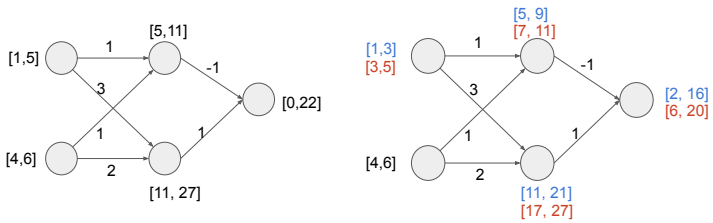


Figure 3.1: Symbolic interval analysis on a toy network. The analysis on the full input space gives an approximation of the reachable output space of $[0, 22]$ while analyzing two subspaces by splitting one input yields a tighter approximation of $[2, 20]$.

[131]: Katz et al. (2017), 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'

[60]: Brix et al. (2024), 'The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results'

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

reachable space by partitioning the input space into a finite number of subspaces.

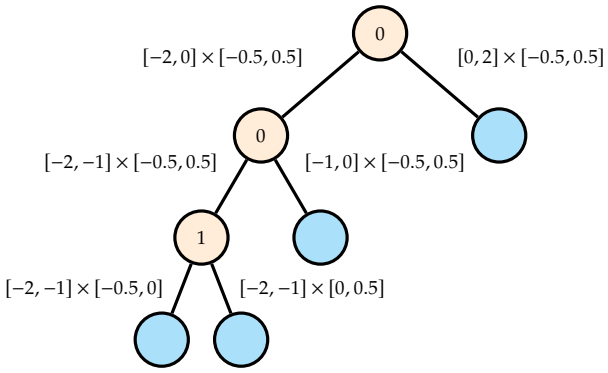
Remark 3.1.1 Note that the convergence of partitioning to the computation of the exact reachable set is not guaranteed for every linearization technique. A condition for the theoretical convergence is **inclusion isotonicity** [58]: if $\mathcal{X}' \subset \mathcal{X}$ then $f_\theta(\mathcal{X}')_{\mathbb{L}} \subset f_\theta(\mathcal{X})_{\mathbb{L}}$ where $f_\theta(\mathcal{X})_{\mathbb{L}}$ is the over-approximation of the reachable space of f_θ applied to $\mathcal{X}$ approximated with the linearization technique $\mathbb{L}$. CROWN / DeepPoly do not satisfy this property as seen in example However, in practice the partitioning of the input space with CROWN / DeepPoly works well and repetitive splits of the input space eventually yield more precise bounds.

Minimizing the number of regions to analyze is important: even for relatively small networks, it may be necessary to divide the input space into thousands of subspaces. For instance, in Singh et al. [112], proving property 9 of the ACAS Xu benchmark requires partitioning into 6300 regions. In this chapter, we focus on minimizing the number of regions by choosing the best input to split at each partitioning step.

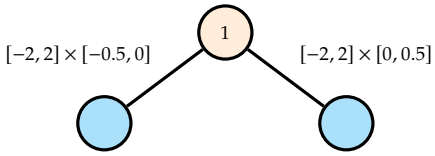
[112]: Singh et al. (2019), 'Boosting Robustness Certification of Neural Networks'

3.1.2 Binary Partitioning Trees

We can represent the partitioning with a tree where each node is a subproblem to solve. When the analysis is inconclusive, an input is chosen and bisected (split at its mid-point), leading to two children nodes. We illustrate the impact of the choice of the input on a fictive toy case with a 2-dimension input: $x \in [-2, 2] \times [-0.5, 0.5]$ in Figure 3.2.



(a) Splitting on $x[0]$ first: 3 splits needed, 7 single passes in total.



(b) Splitting on $x[1]$ first: 3 splits needed, 7 single passes in total.

Figure 3.2: Impact of the choice of the input to split. We indicate the chosen split dimension as a node label and the new input space of the splitted variable on the edges. Cyan indicate when the subproblem is solved, orange when it needs to be split further.

3.2 Heuristics for input partitioning

We now present two baselines, a heuristic from previous work, and our heuristic ReCIPH for choosing which input to split at each partitioning step.

3.2.1 Random choice

A simple heuristic is to choose randomly which input variable to split on. This strategy is used in [132] in a semi-formal framework that estimates output bounds through input sampling. It is also used in Libra [57], a static analyzer for certifying fairness of neural networks. We show in Section 3.4.4 the gains from using ReCIPH in this case. In our experiments, the random choice does not work well and is significantly worse than the following strategy.

[132]: Corsi et al. (2020), 'Evaluating the Safety of Deep Reinforcement Learning Models using Semi-Formal Verification'

[57]: Mazzucato and Urban (2021), 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'

3.2.2 Biggest interval first

Another simple heuristic is to choose the input with the biggest interval to split on first. Intuitively, the larger the interval, the more influence the input should have on the outputs. It requires no additional computation. However, it does not take into account any information from the network or the previous analyses.

3.2.3 Gradient Smears

In Wang et al. [58] another heuristic to choose the best split is proposed: gradient intervals are computed using the bounds on the outputs and the weights of the network. The final score is computed using the highest upper bound of the gradient intervals, multiplied by the width of each input. This heuristic relies on the intuition that the gradients are a good approximation of the influence of the inputs on the outputs. It uses

the weights of the network and the output bounds, but it requires an additional backward pass to compute the gradients.

3.3 ReCIPH: Relational Coefficient for an Input Partitioning Heuristic

We now describe in detail our heuristic, which relies on the relational coefficients obtained at the end of an analysis using a linearization technique

3.3.1 Formalism

For clarity we consider neural networks with a single output and the safety property of checking the sign of the output of the network.

Let f_θ be a neural network with n inputs and a single output, and $\mathcal{X} \subset \mathbb{R}^n$ be the input space of f_θ. Let $\mathcal{I} = [l_0, u_0] \times \cdots \times [l_{n-1}, u_{n-1}]$ be a product of intervals such that $\mathcal{I} \subset \mathbb{R}^n$. We say that f_θ is **safe** on $\mathcal{I}$ if $f_\theta(\mathcal{I}) \geq 0$, *i.e.* the output of the network is non-negative for all inputs in $\mathcal{I}$. For a linearization $\mathbb{L}$, we write $\underline{\Lambda}_\mathbb{L} \in \mathbb{R}^n, \underline{b} \in \mathbb{R}$ and $\overline{\Lambda}_\mathbb{L} \in \mathbb{R}^n, \overline{b} \in \mathbb{R}$ the coefficient vectors and bias obtained by the analysis of the network f_θ on $\mathcal{I}$ using the linear approximation $\mathbb{L}$[2]. Using sound linearizations as described, we have:

2: *a.k.a* the symbolic lower and upper bounds.

$$\forall \boldsymbol{x} \in \mathcal{I}, \underline{\Lambda}_\mathbb{L} \cdot \boldsymbol{x} + \underline{b} \leq f_\theta(\boldsymbol{x}) \leq \overline{\Lambda}_\mathbb{L} \cdot \boldsymbol{x} + \overline{b} \tag{3.1}$$

with the over-approximation of the reachable set of f_θ applied to $\mathcal{I}$ using the linearization technique $\mathbb{L}$.

3.3.2 ReCIPH score

Definition 3.3.1 (ReCIPH score) *We define the **ReCIPH score** of an input variable x_i, denoted as $\rho(x_i)$, ad the magnitude of the corresponding coefficient in the linearization weighted by the width of the input interval:*

$$\rho(x_i) = (u_i - l_i) \cdot \frac{|\underline{\Lambda}_\mathbb{L}[i] + \overline{\Lambda}_\mathbb{L}[i]|}{2}. \tag{3.2}$$

The choice of the interval to bisect is then the one with the highest ReCIPH score:

$$\textit{split axis} = \underset{i \in [0, n-1]}{\operatorname{argmax}} \rho(x_i) \tag{3.3}$$

Intuitively, the coefficients approximate the influence of each input on the output. We weight the magnitude of the final coefficients by the input intervals to account for large disparities between input intervals that could arise from the original specification or from repetitive splits on the same interval. If the model is locally linear, the coefficients are exactly the gradients of the output with respect to the inputs.

Remark 3.3.1 (Extension to multiple outputs) We propose the following heuristic in the case of multiple outputs:

- the property involves a conjunction of comparisons[3]: we simply consider the average ReCIPH score over all outputs and choose the input with the highest average score.
- the property involves a disjunction of comparisons, *e.g.* proving that at least one output is positive: we first use the output bounds to determine which output is the closest to being positive, and then use the ReCIPH scores of that output only.

This assumes that the property holds and thus that we either need to prove all output inequalities (for conjunctions), or that we need to prove one output inequality (for disjunctions). If the property is assumed falsifiable, the heuristic should be swapped: only one inequality needs to be violated to falsify a conjunction, and all inequalities need to be violated to falsify a disjunction. This can be chosen freely by the user in our implementation.

3: This is the case for local robustness assessed on the logits difference: we want to check that every output is positive.

3.4 Experimental results

We evaluate the previous heuristics using the Python Reachability Assessment Tool (PyRAT) [59]. PyRAT is a static analyzer written in Python and relies on computing libraries such as NumPy or PyTorch to verify neural network properties. Several abstract domains are implemented, including the DeepPoly [112] and DeepZono [39] domains. For the gradient smear heuristic we evaluate it directly using ERAN [4], the original tool implementing DeepPoly and DeepZono.

3.4.1 PyRAT on ACAS benchmark

We first evaluate ReCIPH compared to the baseline heuristic of choosing the widest interval and to the gradient smear heuristic re-implemented in PyRAT.

The ACAS Xu networks, introduced in Julian et al. [130] and further described in Julian et al. [133] are feedforward networks with ReLU activations designed to simulate the ACAS X system. The ACAS X system was built to give advisories to pilots to avoid collision with other aircraft. However, the ACAS X system requires the use of a discrete look-up table of over 2GB. The ACAS Xu networks were developed to approximate the ACAS X system without requiring the look-up table. Originally, one network was used with 7 inputs representing the position and velocity of both aircraft and the previous advisory, along with 5 outputs—a score for each possible advisory: Clear-of-Conflict (COC), Weak Left (WL), Weak Right (WR), Strong Left (SL), Strong Right (SR) as illustrated in Figure 3.3. The network had 6 hidden layers of 50 nodes each, giving a total 300 ReLU nodes (not decoupling the activation). Two of the inputs (the previous advisory and the vertical separation between the aircraft) were discretized into 5 ranges each, resulting in 45 networks with 5 inputs each.

Katz et al. [131] introduce a set of 10 properties that the ACAS Xu networks should verify[5].

We present in Table 3.1 the summarized results on all the networks of property 1, 42 networks of property 3 and 4, and property 5, 9 and 10^6 on the ACAS Xu benchmark., analyzed using the DeepPoly abstract domain. They represent a variety of difficulty with some properties

[59]: Lemesle et al. (2024), 'Neural Network Verification with PyRAT'

[112]: Singh et al. (2019), 'Boosting Robustness Certification of Neural Networks'

[39]: Singh et al. (2018), 'Fast and Effective Robustness Certification'

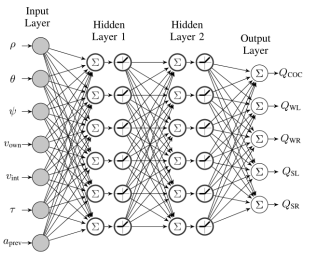
4: `https://github.com/eth-sri/eran`

[130]: Julian et al. (2016), 'Policy compression for aircraft collision avoidance systems'

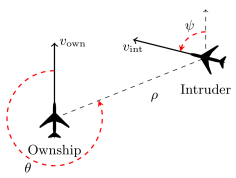
[133]: Julian et al. (2019), 'Deep neural network compression for aircraft collision avoidance systems'

[131]: Katz et al. (2017), 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'

5: For instance property ϕ_3 states that if the intruder is directly ahead and is moving toward the ownship, the score for COC will not be minimal. *e.g.* the network will not advise COC.



(a) ACAS Xu network



(b) ACAS Xu inputs

Figure 3.3: The ACAS Xu networks and its inputs. Credit for both figures: [133].

6: 3 networks do not verify properties 3 and 4. Properties 5, 9 and 10 are only defined for one network in Katz et al. [134]

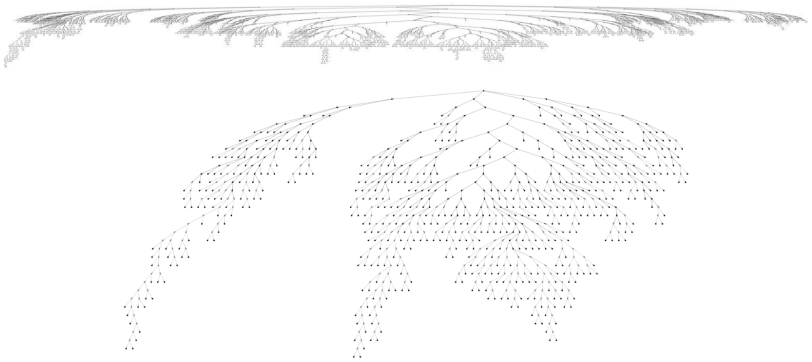


Figure 3.4: We implemented in PyRAT the logging of the binary partitioning trees (see Section 3.1.2). This is an example from the ACAS Xu benchmark, property 4 on network 1_1. Top tree is the analysis using the width heuristic, bottom tree with ReCIPH.

verifiable in fewer than 10 passes and other requiring more than 100 000 passes. We notably exclude properties that are violated: most of them require significantly longer time and we could not run all of them with a reasonable timeout. We did notice similar improvements on a few falsified properties we tested, but overall the input partitioning strategy on its own does not perform well when falsifying properties: an additional search for adversaries is usually used to falsify properties more efficiently.

Compared to splitting on the widest input, ReCIPH significantly improves performance, requiring up to 19 times fewer analyses. Total analysis time is improved proportionally: both heuristics require no additional computation and can be parallelized. We show the binary partitioning trees for property 4 on network 1_1 in Figure 3.4. When comparing ReCIPH to the gradient smear scores, we see that ReCIPH requires significantly fewer splits on properties 3 and 5—13 and 4 times fewer, respectively. For properties 4 and 10, the number of splits using ReCIPH is comparable to the use of the gradient smear scores. Nevertheless, even with a similar number of splits, ReCIPH does not require any extra computation to work and is thus more advantageous.

3.4.2 Overhead of Gradient Smear and ReCIPH

In our experiments, computing the gradient smear scores takes the majority of the analysis time, exceeding the time for the reachability analyses themselves. For example, for network 1_1 on property 4, gradient computation takes more than 75% of total analysis time—39 seconds out of a total of 51 seconds. Even with the default input partitioning in ERAN with the same property and network, the analysis runs in a total of 28 seconds with 19 seconds for gradient computation. In Table 3.1 we compare the time taken to prove the properties 1, 3 and 4 by ReCIPH and the gradient smear heuristic used in ERAN. We use two modes of ERAN:

Table 3.1: Number of one-pass analysis necessary to prove several properties on the ACAS benchmark, using the DeepPoly domain

Property	# Networks	Width	ReCIPH	Gradient Smear
ϕ_1	45	68467	**15289**	16521
ϕ_3	42	319306	**16140**	209924
ϕ_4	42	25906	**1828**	2206
ϕ_5	1	142637	**7023**	29169
ϕ_9	1	3055	**2395**	3877
ϕ_{10}	1	1331	335	**217**

- Iterative bisection, where one input is split at a time, as done in PyRAT with ReCIPH or in Wang et al. [58] with gradient smear scores.
- ERAN default mode, where the initial partition is done by splitting multiple inputs into multiple partitions each according to their gradient smear, forming a queue of hundreds of subproblems to solve. Those problems are then solved with the iterative bisection if necessary[7].

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

7: We did not find the description of the strategy, or the justification for the choices of the size of the partition, in the literature. We described here what we could gather from ERAN's code. We believe the choice of directly splitting multiple inputs is to avoid the overhead of multiple gradient smear computations and the exact size of the partition is chosen empirically.

We run the benchmark on the same machine without parallelization, with the DeepPoly domain for both tools. Unsafe properties are excluded, as ERAN does not provide falsification with the DeepPoly domain without the complete option; in this case, the analysis runs until the timeout is reached.

As we can see, for almost all properties, the default ERAN performs better than bisection. We believe it is due to two main reasons:

- splitting into a finer partition directly requires less time to compute the gradient
- in some cases, it requires fewer subproblems to solve: the "easy" regions are directly proven safe because the partition is finer, whereas multiple splits might be necessary with bisection.

However, splitting into a finer partition might lead to more subproblems if the number of regions is initially too high. Overall ReCIPH still outperforms the default ERAN partitioning strategy.

Table 3.2: Total time in seconds to prove properties 1, 3 and 4 of the ACAS Benchmark. *: We exclude two networks where ERAN times out and report the total time over the remaining networks.

Property	# Networks	ERAN Bisection	ERAN Default	PyRAT ReCIPH
ϕ_1	45	1411.07	370.21	**111.93**
ϕ_3*	40	274.84	100.59	**15.54**
ϕ_4	42	118.06	82.41	**16.86**

3.4.3 PyRAT on Mooring lines monitoring neural network

We also evaluate our heuristic on an industrial partner use-case: neural networks for the detection of mooring line failures on offshore platforms [135, 136]. The network provided is a small fully-connected network with 3 hidden layers of 25 nodes each and with 7 inputs and 5 outputs. The network exists in two versions, one with ReLU activation functions and the other with sigmoid activation functions. Unfortunately, the properties used for this use case are under NDA, and we cannot detail them. They are domain-specific safety properties in the same spirit as the ACAS properties [134]. We first analyze the ReLU activation network on four properties using the DeepPoly and DeepZono domains, with results detailed in Table 3.3. Similar to the ACAS benchmark, we observe a sharp decrease in the number of analysis passes with the DeepPoly domain. These results also hold when using the DeepZono domain, with 3 to 7 times fewer analyses, showing that our heuristic extends beyond the DeepPoly domain.

[135]: Sidarta et al. (2018), 'Damage Detection of Offshore Platform Mooring Line Using Artificial Neural Network'

[136]: Sidarta et al. (2019), 'Detection of Mooring Line Failure of a Spread-Moored FPSO: Part 1 — Development of an Artificial Neural Network Based Model'

[134]: Katz et al. (2017), 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'

Next, we test our heuristic on the sigmoid activation network using only the DeepZono domain on similar properties. As seen in Table 3.4, ReCIPH also greatly decreases the number of passes required even with a sigmoid activation function and its over-approximation.

Table 3.3: Number of one-pass analysis necessary to prove several properties on a 3x25 fully-connected ReLU network, using the DeepPoly and DeepZono domains (marked respectively with "P" and "Z")

Properties	P Width	P ReCIPH	Z Width	Z ReCIPH
1	2897	**843**	3051	879
2	37917	**6565**	53589	9883
3	6515	**993**	10063	1373
4	9105	**1531**	10203	1751

Table 3.4: Number of one-pass analysis necessary to prove two properties on a 3x25 fully-connected Sigmoid network, using the DeepZono domain

Property	Width	ReCIPH
1	147	**5**
2*	1446538	**1855**
3*	timeout	**175137**
4	1613	**23**

* Properties proven false.

3.4.4 Libra on a fairness benchmark

[57]: Mazzucato and Urban (2021), 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'

The implementation of ReCIPH within Libra and the conducted experiments were done by the authors of Libra.

ReCIPH has also been implemented in Libra [57], a static analyzer for certifying fairness of fully-connected neural networks used for classification of tabular data. Libra relies on abstract domains such as DeepPoly, combining a sound forward pre-analysis with an exact backward analysis to quantify bias in a given input space. In this context, a bias is a region of the input space that leads to different classification solely due to a change in value of a sensitive input.

The backward pass complexity is exponential in the number of unstable ReLU nodes. The forward pre-analysis partitions the input space of a ReLU network into **feasible** subspaces by computing a sound over-approximation of the values in each node, and splitting the input space to maximize the number of ReLU nodes with fixed status (either guaranteed to be always active or always inactive). To perform the partitioning in reasonable time, Libra uses an upper bound U on the number of ReLU nodes with non-fixed status and a lower bound L on the size of the input intervals. The partitioning stops when either the number of ReLU nodes with non-fixed status is below U (in which case the region is considered **feasible**) or when the input intervals become too small, falling below L (in which case the region is **unfeasible** and Libra does not proceed with the backward analysis). At the end of the partitioning, we thus obtain a coverage of the original input space, the size of which depends on L and U, the abstract domain used, and the splitting heuristic.

As seen in Table 3.5, using ReCIPH over the default random strategy for the splitting heuristic of Libra greatly improves the coverage of the feasible space. With the same configuration of L and U it can almost double this feasible space and in all cases the new coverage is above 94%.

Table 3.5: Percentage of the feasible space, using the DeepPoly domain in Libra for different configuration of L and U

L, U values	Random	ReCIPH
$L = 0.5,\ U = 3$	49.01%	**94.40%**
$L = 0.5,\ U = 5$	56.15%	**97.03%**
$L = 0.25, U = 3$	81.82%	**99.74%**
$L = 0.25, U = 5$	91.58%	**99.98%**

3.4.5 VNN-COMP Results

We now report results from the 5th International Verification of Neural Networks Competition (VNN-COMP 2024) [60], the latest edition with a public report at the time of writing. PyRAT participated in the edition and was ranked second overall out of eight tools.

[60]: Brix et al. (2024), 'The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results'

The version of PyRAT used in the competition contains many improvements over the version used in Section 3.4.1 and Section 3.4.3, most notably the support of branching on ReLU nodes [59]. However, some of the benchmarks, described in Table 3.7, were analyzed using input partitioning combined with linear approximations and ReCIPH scores[8]. These comprise 6 of the 12 benchmarks in the regular track of the competition, highlighting the practical relevance of our heuristic.

[59]: Lemesle et al. (2024), 'Neural Network Verification with PyRAT'

8: We refer to the VNN-COMP 2024 report [60] for a full description of the benchmarks and participating tools.

For example the cGAN benchmark is a complex benchmark on relatively large networks, applied to image-related tasks. However, the networks' inputs are only 5-dimensional: they take a distance condition (scalar) and a noise vector (4-dimensional) as inputs that are passed to a generator that produces large dimensional outputs (up to 64x64), then processed by a discriminator that outputs a single scalar to be verified (the predicted distance). Similarly, some semantic properties of neural networks used in image classification (e.g., robustness to rotations) can be encoded into a neural network with low-dimensional inputs (e.g., the angle of rotation) [55], making input partitioning a viable refinement solution for verifying such properties.

[55]: Mohapatra et al. (2020), 'Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations'

The ranking of PyRAT on the relevant benchmarks is reported in Table 3.6. Overall, PyRAT with the ReCIPH heuristic performs competitively with other tools.

Benchmark	PyRAT rank	% Verified	% Falsified	Notes
cGAN	1/6	100	100	Equality with α,β-CROWN
LinearizeNN	1/4	100	100	Equality with α,β-CROWN and Marabou
Collins RUL CNN	5/7	100	87.5	The first 4 tools have equal scores.
TLL Verify Bench	1/8	100	100	Equality with CORA and α,β-CROWN
Acas XU	3/8	98.5	100	-
Dist Shift	1/5	-	-	Equality with CORA and α,β-CROWN.

Table 3.6: We report the rank out of the participating tools in each benchmark as well as the number of instances verified (the behavior of the networks satisfied the specification for all the specified input space) and falsified (a counterexample was successfully found). For Dist-Shift all three tools verify and falsify every property except for one where they all timeout (on the same property). As no tool solves every property and no ground-truth is provided we cannot compute coverage of verified / falsified instances but can report a global coverage of 98.6 % of instances solved overall for all three tools.

Table 3.7: VNN-COMP 2024 benchmarks where PyRAT relies on ReCIPH scores and input partitioning.

Category	Benchmark	Application	Network Types	# Params	Effective Input Dim
Complex	cGAN	Image Generation & Image Prediction	Conv. + Vision Transformer	500k - 68M	5
	LinearizeNN	NN controller approximation	FC. + Conv. + Vision Transformer + Residual + ReLU	203k	4
CNN & ResNet	Collins RUL CNN	Condition Based Maintenance	Conv. + ReLU, Dropout	60k - 262k	400 - 800
FC	TLL Verify Bench	Two-Level Lattice NN	Two-Level Lattice NN (FC. + ReLU)	17k - 67M	2
	Acas XU	Collision Detection	FC. + ReLU	13k	5
	Dist Shift	Distribution Shift Detection	FC. + ReLU + Sigmoid	342k - 855k	792

3.5 Related Work

Early works address the verification problem of neural networks by encoding it as an optimization problem solved exactly by a SMT/SAT solver [131, 137–139] or MIP solver [140]. The reliance on complete solvers often makes them impractical at scale. By contrast, bound propagation, as described in Section 2.4.2, yields fast but coarse bounds In this section, we focus on approaches to tighten bounds, either by partitioning the input space or by branching on ReLU nodes, both a form of Branch and Bound (BaB). We conclude by briefly presenting methods that go beyond single-neuron relaxations and methods that tackle network robustness via Lipschitz constants.

[131]: Katz et al. (2017), 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'

[137]: Pulina and Tacchella (2012), 'Challenging SMT solvers to verify neural networks'

[138]: Ehlers (2017), 'Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks'

[139]: Huang et al. (2016), 'Safety Verification of Deep Neural Networks'

[140]: Tjeng et al. (2019), 'Evaluating Robustness of Neural Networks with Mixed Integer Programming'

ReCIPH like heuristics The work of Henriksen and Lomuscio [100] uses the parallel linear approximation, with one linear equation and an error matrix to compute concrete bounds for each node, akin to a zonotope approach. The errors are also used to score ReLU nodes to split on. The approach is similar to ours, with the error matrix capturing some relations between layers, and readily available after bound propagation. However, they use to estimate the influence of ReLU nodes rather than inputs, and it is limited to their parallel linear approximation. A concurrent work of ours, presented at the same workshop, Kern et al. [141], builds on the same approach but uses symbolic interval analysis with fresh variables, enabling nonparallel linear approximations. They also perform input partitioning, but their split score is tailored to ReLU networks: instead of input-output relations, they average the coefficients linking unstable internal ReLU nodes to the inputs. The goal is to split the input that most affects ReLU activation states rather than the output bounds. More recently, Koller et al. [142] also partition the input space to limit the number of unstable ReLUs. Implemented in CORA [143], they use constrained zonotopes to model post-split regions rather than products of intervals, which allows them to exploit constraints derived from ReLU status in latent space. Finally, α,β-CROWN has used a ReCIPH-like input-splitting heuristic, **sb-fast**, since its September 2022 release, although we have not seen it described in the literature.

[100]: Henriksen and Lomuscio (2021), 'DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis'

[141]: Kern et al. (2022), 'Optimized symbolic interval propagation for neural network verification'

[142]: Koller et al. (2025), 'Out of the Shadows: Exploring a Latent Space for Neural Network Verification'

[143]: Althoff (2015), 'An Introduction to CORA 2015'

Several works compute tighter bounds by dividing the verification problem into smaller subproblems based on ReLU states, another form of Branch and Bound. If 0 lies within a ReLU node's pre-activation bounds, the problem can be split into two subproblems by considering the positive and negative input cases in separate bounding problems. In these works, bound propagation and linear approximations are used to first estimate node status, while post-split bounds are computed with more precise methods.

Branch and Bound with Off-the-shelf Solvers Neurify [111] uses linear approximations (without backsubstitution) to compute intermediate bounds and prioritizes nodes using the gradient-smear heuristic. After the split, the bounding problem is solved with a Linear Programming (LP) solver. Royo et al. [144] builds on Neurify using a different prioritization strategy based on shadow prices, a form of sensitivity analysis relying on the LP formulation. Inputs are split according to their estimated influence on ReLU node status. VeriNet [100, 120], discussed in the previous paragraph, use a different LP-encoding, a gradient based counterexample search and a different view of the parallel linear approximation. Their method is also adapted to other activations such as sigmoid or tanh. Bak et al. [145] and Singh et al. [112] both use parallel linear approximations in the form of zonotopes before splitting ReLU nodes. Singh et al. [112] also

[111]: Wang et al. (2018), 'Efficient Formal Safety Analysis of Neural Networks'

[144]: Royo et al. (2019), 'Fast Neural Network Verification via Shadow Prices'

[120]: Henriksen and Lomuscio (2020), 'Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search'

[145]: Bak et al. (2020), 'Improved Geometric Path Enumeration for Verifying ReLU Neural Networks'

[112]: Singh et al. (2019), 'Boosting Robustness Certification of Neural Networks'

mixes the use of LP and MIP formulations, with a heuristic to choose when to use either one. Bak et al. [145] use Star Sets [146], a method allowing approximate bound computation using the convex hull relaxation of ReLU activations, and exact analysis via branch and bound.Katz et al. [147] improves upon Katz et al. [131] with a specialized SMT solver, uses DeepPoly relaxations, and employs a heuristic based on pre-activation bounds to choose ReLU nodes to split on. Bunel et al. [148] use both bound propagation and an LP solver to compute intermediate bounds, depending on the heuristic. They consider either the largest input interval heuristic (**BaB-input**) or estimate the impact of splitting by computing approximate final bounds using parallel linear approximations for each subproblem considered (**Smart Branching, BaBSB**). While BaBSB is an input partitioning strategy it is most beneficial when the subproblem is later solved with a more precise and costly method. They also split on ReLU nodes, prioritizing them with another scoring heuristic (**Smart ReLU, BaBSR**) that estimates influence via a single backward pass. Rather than using hand-crafted heuristics, Lu and Kumar [149] formulate branching as a learning problem and use a graph neural network to prioritize which nodes to split on.

Efficient Parallelised Branch and Bound Dvijotham et al. [150] relies on duality and a Lagrangian relaxation of the verification problem, introducing a customized solver as an alternative to off-the-shelf LP-solvers. While the approach is still incomplete, it can be made complete by combining it with branch and bound as in Wang et al. [151] and De Palma et al. [152]. Both of these work introduce different relaxations, and use a fast parallelizable implementation with supergradient methods on a GPU. This is a major step toward scalable branch and bound: the loss of precision relative to the LP formulation is largely compensated by the speedup, bounding and pruning more subproblems faster in the branching process. De Palma et al. [152] also proposes a new branching heuristic (**Fast Smart Branching, FSB**), a method more costly than BaBSR, ranking ReLU nodes with a fast scoring heuristic then computing fast bounds only on the best candidates to further refine the ranking. This approach remains the key ingredient to the state of the art solver α,β-CROWN, which also comprises Branch and Bound counterexample search [153], extensions to general non-linearities [123], support of general cutting planes, additional linear constraints not involved in the LP formulation that can tighten the bounds [154, 155]. MN-BAB [41] similarly uses a Lagrangian formulation and GPU parallelization for BaB, but it is based on a multi-neuron approximation of Singh et al. [156] and uses a priority heuristic adapted to the multi-neuron case.

Tighter relaxations Salman et al. [115] showed that any bounding method relying on the triangle relaxation [126] (illustrated in Figure 2.7), faces an inherent barrier limiting the tightness of the computed bounds [9]. Orthogonal to improvements obtained from branching on ReLU nodes, a line of work seeks to overcome this barrier by employing tighter relaxations. Singh et al. [156] proposes to formulate a multi-neuron relaxation (*kPoly*), considering multiple ReLU nodes within the same layer. Anderson et al. [157] keeps the focus on single activation node but include the preceding affine layer of the activation when building the convex hull. This convex relaxation is tighter than the triangle relaxation but can introduce an exponential number of constraints in the worst case. Tjandraatmadja et al. [158] improves on the formulation and propose a specialized cutting plane method to compute the bounds. De Palma et al. [159], building on previous work [160], uses a similar relaxation, dual

[146]: Tran et al. (2019), 'Star-Based Reachability Analysis of Deep Neural Networks'

[147]: Katz et al. (2019), 'The Marabou Framework for Verification and Analysis of Deep Neural Networks'

[148]: Bunel et al. (2020), 'Branch and Bound for Piecewise Linear Neural Network Verification'

[149]: Lu and Kumar (2020), 'Neural Network Branching for Neural Network Verification'

[150]: Dvijotham et al. (2018), 'A dual approach to scalable verification of deep networks'

[151]: Wang et al. (2021), 'Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification'

[152]: De Palma et al. (2021), *Improved Branch and Bound for Neural Network Verification via Lagrangian Decomposition*

[153]: Zhang et al. (2022), 'A Branch and Bound Framework for Stronger Adversarial Attacks of ReLU Networks'

[123]: Shi et al. (2025), 'Neural Network Verification with Branch-and-Bound for General Nonlinearities'

[154]: Zhang et al. (2022), 'General Cutting Planes for Bound-Propagation-Based Neural Network Verification'

[155]: Zhou et al. (2024), 'Scalable Neural Network Verification with Branch-and-bound Inferred Cutting Planes'

[41]: Ferrari et al. (2022), 'Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound'

[156]: Singh et al. (2019), 'Beyond the single neuron convex barrier for neural network certification'

[115]: Salman et al. (2019), 'A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks'

[126]: Elboher et al. (2019), 'An Abstraction-Based Framework for Neural Network Verification'

9: Bounds computed with the linear approximations discussed in Section 2.4.2 or by off-the-shelf LP-solvers fall in this category.

[157]: Anderson et al. (2020), 'Strong mixed-integer programming formulations for trained neural networks'

[158]: Tjandraatmadja et al. (2020), 'The Convex Relaxation Barrier, Revisited: Tightened Single-Neuron Relaxations for Neural Network Verification'

[159]: De Palma et al. (2024), 'Scaling the Convex Barrier with Sparse Dual Algorithms'

[160]: De Palma et al. (2021), 'Scaling the Convex Barrier with Active Sets'

solvers and supergradient methods to further improve the scalability of the approach. Another approach relaxes the verification problem as a semidefinite program (SDP). Raghunathan et al. [2] first introduce linear and quadratic constraints on the ReLU activations then further relax them into a SDP. This allows to constrain ReLU nodes jointly. Dathathri et al. [161] generalizes this formulation using its dual, making it applicable beyond ReLU networks, and solve it with subgradient methods on GPUs. Batten et al. [3] adds linear cuts from the triangle relaxation to the SDP formulation, leading to a provably tighter relaxation. They rely on a layer-based decomposition method to efficiently solve the SDP. We illustrate the approach in Figure 3.5. Recently, Chiu et al. [162] instead build on SDP relaxations to improve linear bound propagation for ℓ_2 robustness certification. Owing to their cubic complexity [162], SDP relaxations are generally not as scalable as BaB with LP-based methods: In the time taken to solve a single SDP, BaB methods can solve many subproblems using an LP formulation, leading to tighter bounds.

[2]: Raghunathan et al. (2018), 'Semidefinite relaxations for certifying robustness to adversarial examples'

[161]: Dathathri et al. (2020), 'Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming'

[3]: Batten et al. (2021), 'Efficient Neural Network Verification via Layer-based Semidefinite Relaxations and Linear Cuts'

[162]: Chiu et al. (2025), 'SDP-CROWN: Efficient Bound Propagation for Neural Network Verification with Tightness of Semidefinite Programming'

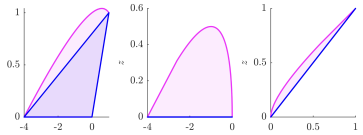


Figure 3.5: SDP and LP relaxations of a piecewise linear activation. The SDP relaxation of Raghunathan et al. [2] is delimited in pink, the triangle relaxation in blue. The unstable case is left, the inactive case in the middle and active case to the right. Notice that the SDP relaxation is not exact even in the fixed status, motivating the introduction of linear cuts to the SDP. Figure from Batten et al. [3].

Lipschitz constants Finally, verification can also be tackled by estimating a network's Lipschitz constant—a bound on how much the output can change when the input changes. Bound propagation methods can be used for this purpose [163] as well as SDP relaxations [164]. Specialized network architectures can also be designed to have low Lipschitz constants, enabling certified robustness by construction [165–169].

[163]: Shi et al. (2022), 'Efficiently computing local lipschitz constants of neural networks via bound propagation'

[164]: Fazlyab et al. (2019), 'Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks'

[165]: Zhang et al. (2021), 'Certified Robustness to Programmable Transformations in LSTMs'

[166]: Zhang et al. (2022), 'Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective'

[167]: Anil et al. (2019), 'Sorting out Lipschitz function approximation'

[168]: Araujo et al. (2023), 'A Unified Algebraic Perspective on Lipschitz Neural Networks'

[169]: Leino et al. (2021), 'Globally-robust neural networks'

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

3.6 Conclusion

In this chapter, we showed that, using coefficients from linear approximations to select the best input variable to split, greatly reduces the number of regions to analyze compared to splitting on the largest input interval first. The strategy also requires no extra computation (except possibly averaging the coefficients). Thanks to this, ReCIPH outperforms the gradient-smear heuristic [58] even when both use similar linear approximations, since each split with gradient smear requires an additional backward pass to compute the scores.
Overall, we showed that our heuristic is not limited to a single abstract domain, activation function, or tool. ReCIPH readily extends to different linear approximations and tools with similar gains.
The main limitation of ReCIPH is its applicability primarily to neural networks with low-dimensional inputs. Other methods, such as the branch-and-bound approaches discussed in Section 3.5, are more appropriate for high-dimensional cases. We highlight that this limit does not make the approach irrelevant as use-cases with low-dimensional inputs remain common.
One possible extension is to accelerate splitting by partitioning more than one input at a time or by splitting inputs into more than two subintervals. Experimentally we see that it can improve analysis time for some properties, but we did not find generalizable heuristics. We also supervised a summer internship to explore a learning approach to find optimal splitting choices. The intern, Thomas Boulanger, attempted a reinforcement-learning approach to learn a splitting strategy. The results were inconclusive: the learned strategies did not consistently lead to fewer subproblems and incurred overhead. Finding an automated, accelerated splitting strategy remains an open problem. With the success of branch and bound methods we do not see the search of automated splitting strategies as a priority, but it could be useful in some specific cases.

Applications of Certified Training

Certified Training for Empirical Robustness

4

In this chapter we present a study on the use of certified training methods to improve the empirical robustness of neural networks.
Adversarial training is arguably the most popular way to provide empirical robustness against specific adversarial examples. While variants based on multi-step adversarial attacks incur significant computational overhead, single-step variants are vulnerable to a failure mode known as catastrophic overfitting, which hinders their practical utility for large perturbations. A parallel line of work, certified training, has focused on producing networks amenable to formal guarantees of robustness against any possible attack. However, the wide gap between the best-performing empirical and certified defenses has severely limited the applicability of the latter.
Inspired by recent developments in certified training, which rely on a combination of adversarial attacks with network over-approximations [8], and by the connections between local linearity and catastrophic overfitting [5, 91], we present experimental evidence on the practical utility and limitations of using certified training toward empirical robustness. We show that, when tuned for the purpose, a recent certified training algorithm can prevent catastrophic overfitting on single-step attacks, and that it can bridge the gap to multi-step baselines under appropriate experimental settings. Finally, we present a conceptually simple regularizer for network over-approximations that can achieve similar effects while markedly reducing runtime.
This is joint work with Alessandro De Palma and led to a publication in the Transactions on Machine Learning Research [62].

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[5]: Andriushchenko and Flammarion (2020), 'Understanding and Improving Fast Adversarial Training'

[91]: Ortiz-Jimenez et al. (2023), 'Catastrophic overfitting can be induced with discriminative non-robust features'

[62]: De Palma et al. (2025), 'On Using Certified Training towards Empirical Robustness'

4.1 Motivation and methodology

Recent certified training techniques (Section 2.3) have demonstrated that the ability to precisely control the tightness of network over-approximations while preserving an adversarial training component is crucial to maximize certified robustness across experimental setups. We hypothesize that this versatility can be alternatively leveraged toward improving the empirical robustness of the adversarial training schemes on top of which they are applied, providing experimental evidence in Section 4.3 and Section 4.4. This section details the certified training algorithms employed in the experimental study. We begin with a motivating example (Section 4.1.1), then study the qualitative behavior of existing methods in settings of interest (Section 4.1.2), and conclude by presenting a conceptually simple regularizer designed to mimic the effect of IBP-based training while reducing the associated overhead (Section 4.1.3).

4.1.1 Motivating Example

In order to motivate our empirical study, we show that multi-step adversarial attacks, which typically exhibit superior robustness at the expense of training time, are associated to smaller network over-approximations, as measured by the IBP loss. Figure 4.1, which is associated to the experimental setup from Table 4.2, shows that the IBP loss indeed decreases with the number of attack steps. Furthermore, it shows two distinct

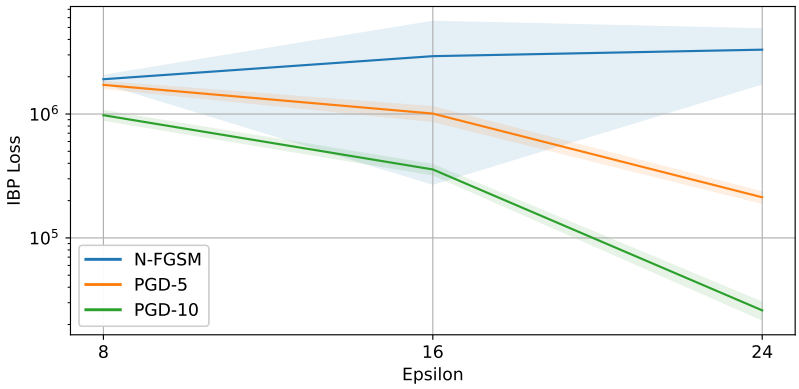


Figure 4.1: IBP loss of adversarial training schemes on CIFAR-10, setup from Table 4.2.

qualitative trends: for N-FGSM, which is prone to CO (causing the large experimental variability), the IBP loss increases with the perturbation radius. On the other hand, $\mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)$ decreases with ϵ for PGD-5 and PGD-10. Indeed, multi-step adversarial attacks yield networks that are more locally linear compared to single-step attacks [5, 7, 91], which in turn results in tighter $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ bounds and hence smaller network over-approximations. In this work, we investigate whether directly controlling $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ via certified training can benefit empirical robustness.

We recall some notations introduced in Section 2.2.3:
$\boldsymbol{z}_{f_\theta}(\boldsymbol{x}, y) := f_\theta(\boldsymbol{x})[y]\mathbf{1} - f_\theta(\boldsymbol{x})$ is the vector of logit differences between the class y and all other classes, and $\boldsymbol{z}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ its element-wise minimum value within the perturbation set $B(\boldsymbol{x}, \epsilon)$. We note $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ a lower bound on $\boldsymbol{z}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$, that is soundly approximated via IBP in this chapter. Given such bounds the IBP loss is defined as $\mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) := \mathcal{L}(-\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}; y)$, where $\mathcal{L}$ is the cross-entropy loss.

[5]: Andriushchenko and Flammarion (2020), 'Understanding and Improving Fast Adversarial Training'

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'

[91]: Ortiz-Jimenez et al. (2023), 'Catastrophic overfitting can be induced with discriminative non-robust features'

4.1.2 Expressive Losses

We recall the definition of the expressive losses defined on different convex combinations parametrized by $\alpha \in [0, 1]$, as described in Example 2.3.1:

$$\mathcal{L}_{\alpha,MTL-IBP} := (1-\alpha) \cdot \mathcal{L}_{\text{adv}}(f_\theta, B(\boldsymbol{x}, \epsilon); y) + \alpha \cdot \mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y).$$
$$\mathcal{L}_{\alpha,EXPIBP} := \mathcal{L}_{\text{adv}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)^{1-\alpha} \cdot \mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x}, \epsilon); y)^{\alpha}.$$
$$\mathcal{L}_{\alpha,CC-IBP} := \mathcal{L}\left(-\left[(1-\alpha) \cdot \boldsymbol{z}_{f_\theta}(\boldsymbol{x}_{\text{adv}}, y) + \alpha \cdot \underline{\mathbf{z}}_{f_\theta}(\boldsymbol{x}, y)\right]; y\right).$$

We refer to Section 2.3.1 and Figure 2.3 for the description of the SABR loss [105], a more involved way to combine adversarial and IBP training.

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[105]: Müller et al. (2023), 'Certified Training: Small Boxes are All You Need'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[104]: Mao et al. (2023), 'TAPS: Connecting Certified and Adversarial Training'

Expressive losses in the certified robustness context In order to maximize certified accuracy (defined as the share of inputs verified to be robust) and stabilize training, previous work [8, 105] deployed the losses under a specific set of conditions. First, shallow architectures, such as the popular 7-layer `CNN-7` [8, 98, 104, 105], which, aided by specialized initialization and regularization [98], limits the growth of the IBP bounds over consecutive layers. Second, relatively long training schedules featuring gradient clipping, a warm-up phase of standard training, and a ramp-up phase during which the perturbation radius used to compute both the attack and the IBP bounds is gradually increased from 0 to the target value ϵ. Finally, relatively large values of α to reduce the magnitude of the bounds $\underline{\mathbf{z}}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ on the logit differences. In this context, all the expressive losses from Example 2.3.1 display a similar qualitative behavior, attaining roughly the same certified robustness [8].

[61]: Shafahi et al. (2019), 'Adversarial training for free!'

Qualitative differences in adversarial training setups In adversarial training, overhead is often a major concern [61]. Indeed, works on single-

step adversarial training typically rely on deeper networks and shorter training schedules to maximize empirical robustness while minimizing runtimes.

A common setting, involves a `PreactResNet18` [75] trained with a cyclic learning rate and without any gradient clipping, ramp-up [1] nor warm-up [4–6].

Owing to the exploding IBP loss at initialization and to the specific training hyperparameters, pure IBP training is not directly applicable to this setup.

In Table 4.1 we compute the IBP loss at initialization on the three network architectures employed in this work, which provides an indication of the size of the IBP network over-approximation. Specifically, we measure the average IBP loss over the standard CIFAR-10 training set, against perturbations of radius $\epsilon = 24/255$. It shows that, as expected, the network bounds explode with the model depth, with `PreActResNet18` displaying an IBP loss at initialization that is several orders of magnitude larger than those of the two employed `CNN` architectures.

Finally, `CNN-7` features an IBP loss almost two orders of magnitude larger than `CNN-5`, explaining the qualitative differences in behavior from Figure 4.12.

In order to study the behavior of expressive losses in this context, we tuned the expressive losses to maximize IBP-based verifiability on CIFAR-10 under perturbations of radius $\epsilon = 8/255$, using N-FGSM to generate $\boldsymbol{x}_{\text{adv}}$. Figure 4.2 shows that MTL-IBP, CC-IBP as well as SABR are unable to attain non-negligible certified accuracy via IBP in this context. On the other hand, Exp-IBP reaches almost 20% IBP certified accuracy, highlighting a significant qualitative difference in behavior.

[75]: He et al. (2016), 'Identity mappings in deep residual networks'

1: Except for the SVHN dataset [170]. Jorge et al. [4] for instance increase the perturbation for the first 5 epochs and show several settings where the trained classifier is constant (a form of underfitting).

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

[80]: Rice et al. (2020), 'Overfitting in adversarially robust deep learning'

Table 4.1: IBP loss at initialization for the network architectures considered in this work, computed on the CIFAR-10 training set against perturbations of radius $\epsilon = 24/255$ (means and standard deviations for 5 runs).

Architecture	IBP loss
`PreActResNet18`	$(2.20 \pm 0.06) \times 10^{16}$
`CNN-7`	$(8.67 \pm 0.18) \times 10^{5}$
`CNN-5`	$(1.11 \pm 0.01) \times 10^{4}$

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

Sensitivity of expressive losses In order to gain further insight on this, we plot the sensitivity of the four expressive losses on two toy networks where the magnitude of the IBP bounds is controlled through a scalar parameter w.

The toy neural networks are fully connected networks of depth n designed to feature IBP bounds which are tunable in magnitude through a scalar parameter w, and explode with the depth of the network. They are defined as follows:

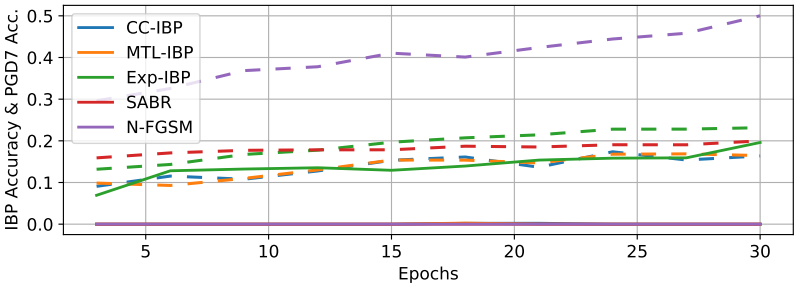
$$\begin{aligned} \boldsymbol{x}^1 &= \text{ReLU}\left(\begin{bmatrix} w & -w \\ -w & w \end{bmatrix} \boldsymbol{x}^0\right), \\ \boldsymbol{x}^n &= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \boldsymbol{x}^{n-1} + \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \\ \boldsymbol{x}^k &= \text{ReLU}\left(\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \boldsymbol{x}^{k-1}\right) \qquad \forall\, k \in [\![2, n-1]\!], \end{aligned} \tag{4.1}$$

evaluating to:

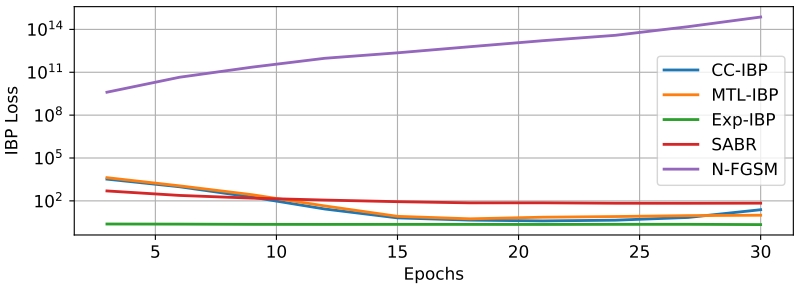
$$\boldsymbol{x}^n = 2^{n-1}\,\text{ReLU}\left(w \begin{bmatrix} \boldsymbol{x}^0[0] - \boldsymbol{x}^0[1] \\ \boldsymbol{x}^0[1] - \boldsymbol{x}^0[0] \end{bmatrix}\right) + \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

We recall notations: upper script to enumerate through layers ($\boldsymbol{x}^k$ is the k-th layer output), $\boldsymbol{x}[i]$ to denote the i-th component of a vector $\boldsymbol{x}$.

Let us assume that $y = 1$ (a similar reasoning holds if $y = 0$). In this



(a) Certified accuracy via IBP bounds (solid lines), empirical accuracy under PGD-7 attacks (dashed).



(b) IBP loss.

Figure 4.2: IBP certified robustness attained by expressive losses on the `PreActResNet18` training setup from Jorge et al. [4]. Validation results on CIFAR-10 under perturbations of $\epsilon = 8/255$. Hyperparameters are: $\alpha = 0.1$ for Exp-IBP, $\alpha = 10^{-15}$ for CC-IBP and MTL-IBP, $\alpha = 10^{-9}$ for SABR. They are chosen to reduce the IBP loss as much as possible on the validation set. N-FGSM is

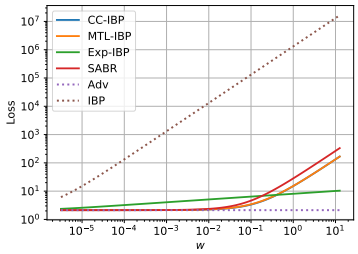
context, if $w \geq 0$, the logit differences $z_{x^n}(x^0, y)$ can be computed as:

$$\begin{aligned} z_{x^n}(x^0, 1) &= 2^{n-1}\left[\text{ReLU}\left(w\left(x^0[1] - x^0[0]\right)\right) - \text{ReLU}\left(w\left(x^0[0] - x^0[1]\right)\right)\right] - 2 \\ &= w\, 2^{n-1}\left[\text{ReLU}\left(x^0[1] - x^0[0]\right) - \text{ReLU}\left(x^0[0] - x^0[1]\right)\right] - 2 \\ &= w\, 2^{n-1}\left(x^0[1] - x^0[0]\right) - 2. \end{aligned}$$

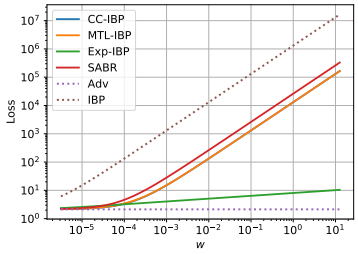
If an $\tilde{x}^0$ such that $\tilde{x}^0_1 - \tilde{x}^0_0 < 0$ belongs to the perturbation set $B(x^0, \epsilon)$, then $z_{x^n}(\tilde{x}^0, 1)$ will have $(-\infty, -2]$ as image for $w \in [0, +\infty)$. Noting that the relative IBP lower bounds $l_{x^n}^{B(x^0,\epsilon),1}$ evaluate to -2 if $w = 0$ (see Eq. (2.16)), that $l_{x^n}^{B(x^0,\epsilon),1} \leq z_{x^n}(\tilde{x}^0, 1)$ for any $w \geq 0$, and that $l_{x^n}^{B(x^0,\epsilon),1}$ is a continuous function with respect to w (as it is computed through compositions and linear combinations of continuous functions), the image of $l_{x^n}^{B(x^0,\epsilon),1}$ for $w \in [0, +\infty)$ will also be $(-\infty, -2]$. In addition, for any fixed $w > 0$, $l_{x^n}^{B(x^0,\epsilon),1}$ will decrease at least as fast as $z_{x^n}(\tilde{x}^0, 1)$ (which decreases exponentially) with the depth of the network.
In order to meet the above condition, Figure 4.3 uses $x^0 = [-5, 5]^T$, $y = 1$ and $B(x^0, 10)$ as perturbation. Furthermore, in order to keep the adversarial loss constant with w for the purposes of Figure 4.3, we set $x_{\text{adv}} = [0, 0]^T$, for which $z_{x^n}(x_{\text{adv}}, 1) = -2$ regardless of w.
Figure 4.3 shows that on the deeper network the MTL-IBP loss either flattens onto $\mathcal{L}_{\text{adv}}(f_\theta, B(x, \epsilon); y)$ at large IBP loss values (small α), or explodes with the IBP bounds, resulting in unstable training behavior (larger α). CC-IBP and SABR display a similar behavior.
On the other hand, Exp-IBP can be tuned so as to be able to drive $\mathcal{L}_{\text{IBP}}(f_\theta, B(x, \epsilon); y)$ to very small values without exploding at larger w values. On the shallow network, instead, for which the IBP bounds are significantly smaller, MTL-IBP, CC-IBP and SABR can be tuned to display roughly the same behavior as Exp-IBP for smaller $\mathcal{L}_{\text{IBP}}(f_\theta, B(x, \epsilon); y)$ without taking overly large loss values within the considered parameter range. We believe this explains their homogeneous behavior in training setups designed to prevent large IBP bounds, such as those typically employed in the certified training literature.

Considering the similar qualitative behavior of MTL-IBP, CC-IBP and SABR, we focus on MTL-IBP and Exp-IBP in the rest of this chapter.

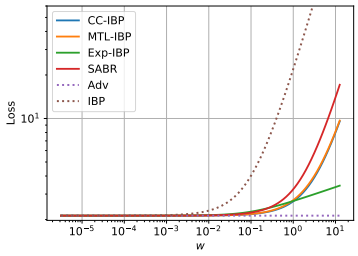
Maximizing empirical accuracy As seen in Figure 4.2, expressive losses can pay a large price in empirical accuracy compared to $\alpha = 0$ (pure N-FGSM, in this case) when α is tuned to tighten the IBP bounds. One may hence conclude that certified robustness is at odds with empirical accuracy. In section Section 4.4 we instead show that, when α is tuned for the purpose, expressive losses can result in increased empirical robustness.



(a) 18 layers; $\alpha = 10^{-5}$ for CC-IBP, MTL-IBP and SABR.



(b) 18 layers; $\alpha = 10^{-2}$ for CC-IBP, MTL-IBP and SABR.



(c) 2 layers; $\alpha = 3 \times 10^{-2}$ for CC-IBP, MTL-IBP and SABR.

Figure 4.3: Sensitivity of the expressive losses on a toy network of varying depth, with $\alpha = 10^{-1}$ for Exp-IBP. For all three plots, CC-IBP displays almost identical behavior to MTL-IBP.

4.1.3 ForwAbs

Expressive losses require the computation of IBP bounds $\boldsymbol{l}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$ using the procedure in Eq. (2.16), whose cost roughly corresponds to two network evaluations (also called forward passes): one using the original network, the other employing the absolute value of the network weights. This overhead on top of adversarial training is negligible only when $\boldsymbol{x}_{\text{adv}}$ are computed via multi-step attacks. We here study less expensive yet conceptually simple ways to control the IBP bounds when single-step attacks are instead employed.

We denote by $\boldsymbol{\delta}^k := (\hat{\boldsymbol{u}}^k - \hat{\boldsymbol{l}}^k)$ the gap between the lower and upper bounds after the k-th affine layer.

For ReLU networks, $\boldsymbol{\delta}^k$ can be upper bounded by $|W^k|\boldsymbol{\delta}^{k-1} \geq |W^k|(\sigma(\hat{\boldsymbol{u}}^{k-1}) - \sigma(\hat{\boldsymbol{l}}^{k-1}))$. This upper bound coincides with $\boldsymbol{\delta}^k$ in the case of Deep Linear Networks (DLNs), for which $\sigma(\boldsymbol{a}) = \boldsymbol{a}$, or if the ReLUs are passing ($\hat{\boldsymbol{l}}^k \geq 0$). We propose to employ $\bar{\boldsymbol{\delta}}^n$, which we define as the (looser) upper bound to $\boldsymbol{\delta}^n$ obtained through repeated use of the $|W^k|\boldsymbol{\delta}^{k-1}$ bound, as a regularizer for the IBP loss. The $\bar{\boldsymbol{\delta}}^n$ term can be computed at the cost of a single forward pass through an auxiliary network with $\sigma(\boldsymbol{a}) = \boldsymbol{a}$ where each affine layer $\mathbf{g}^k(\boldsymbol{a}) = W^k\boldsymbol{a} + \boldsymbol{b}$ is replaced by $\mathbf{h}^k(\boldsymbol{a}) = |W|^k\boldsymbol{a}$. We call the resulting method ForwAbs (*Forw*ard pass in *Abs*olute value).

Definition 4.1.1 *The ForwAbs loss takes the following form:*

$$\mathcal{L}_\lambda^{ForwAbs}(f_\theta, \mathcal{B}_\epsilon(\boldsymbol{x}); y) := \mathcal{L}_{adv}(f_\theta, B(\boldsymbol{x},\epsilon); y) + \lambda \left(\mathbf{1}^T \bar{\boldsymbol{\delta}}^n\right), \quad \textit{where: } \bar{\boldsymbol{\delta}}^1 = \boldsymbol{\delta}^1 = 2\epsilon|W^1|\mathbf{1}, \quad \bar{\boldsymbol{\delta}}^k = |W^k|\bar{\boldsymbol{\delta}}^{k-1} \ \forall\, k \in [\![2, n]\!]. \tag{4.2}$$

In order to demonstrate the empirical correlation between the ForwAbs term $\bar{\boldsymbol{\delta}}^n$ and the IBP bounds $\boldsymbol{l}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$, Figure 4.4 presents the results of a ForwAbs-trained network, where the λ coefficient is tuned to minimize the IBP bounds in the same setup as Figure 4.2. In spite of its simplicity and of its significantly smaller runtime overhead, ForwAbs can drive the IBP loss to roughly the same values attained by MTL-IBP. Furthermore, the final $\mathcal{L}_{\text{IBP}}(f_\theta, B(\boldsymbol{x},\epsilon); y)$ attained by ForwAbs is significantly lower than those associated to N-FGSM. Strong ℓ_1 regularization over the weights (with regularization coefficient $\lambda_{\ell_1} = 0.04$) fails to achieve a comparable effect, demonstrating that the behavior of ForwAbs is not merely a consequence of small ℓ_1 norms of the network weights.

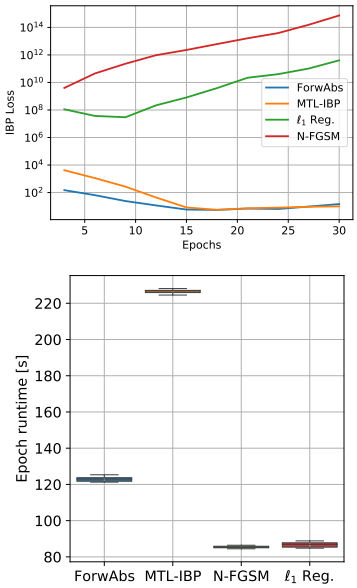


Figure 4.4: IBP loss over epochs (*top*), box plots (10 runs) for the training time of an epoch (*bottom*), setup as Figure 4.2. Hyperparameters are: $\alpha = 0.1$ for ExpIBP, $\alpha = 10^{-15}$ for CC-IBP and MTL-IBP, $\alpha = 10^{-9}$ for SABR, $\tilde{\lambda} = 10^{-15}$ for ForwAbs and $\lambda_{\ell_1} = 0.04$ for ℓ_1-regularized N-FGSM. They are chosen to reduce the IBP loss as much as possible on the validation set. For all methods in this experiment, larger values among those we considered led to numerical problems or trivial behaviors, such as networks consistently outputting the same class in our implementation.

4.2 Experimental Setup

We first present the experiment setup and design choices made for our empirical study.

4.2.1 Datasets

We focus on three standard 32×32 image classification datasets: CIFAR-10 and CIFAR-100 [171], and SVHN [170]. CIFAR-10 and CIFAR-100 consist of 60,000 32×32 RGB images, with 50,000 images for training and 10,000 for testing. CIFAR-10 contains 10 classes, while CIFAR-100 contains 100 classes. SVHN consists of 73,257 32×32 RGB images, with 73,257 images for training and 26,032 for testing.
Unless specified otherwise, for tuning purposes or when reporting validation results we use a random 20% holdout of the training set as validation set, and train on the remaining 80%. After tuning and when reporting test set results, we use the standard train and test splits for all datasets.

4.2.2 Implementation Details

Our implementation relies on PyTorch [124] and on the public codebases from Jorge et al. [4], Rocamora et al. [7], and De Palma et al. [8]. We compute IBP bounds using the Auto_LiRPA implementation [46].
N-FGSM may return a point outside the set of allowed perturbations $B(\boldsymbol{x}, \epsilon)$. As a result, when using it to compute the adversarial point $\boldsymbol{x}_{\text{adv}}$ for SABR, we disable the projection of the SABR perturbation subset onto the original perturbation set, simply setting $\boldsymbol{x}_\alpha = \boldsymbol{x}_{\text{adv}}$, which allows the implementation to meet the definition of expressivity (see Definition 2.3.1) for small α values.
Consistently with the single-step adversarial training literature [4, 7], BatchNorm layers [76], which are present in all the employed models, are always kept in training mode throughout training (including during the attacks). When computing IBP bounds for expressive losses and ForwAbs terms at training time, we use the batch statistics from the current adversarial attack. For ForwAbs, these are retrieved from the Auto_LiRPA implementation [46]. The clean inputs are never fed to the network at training time, except when using FGSM to generate the attack or when training with pure IBP (in that case, the clean batch statistics are used to compute the IBP bounds). The outcome of the running statistics compute during training is systematically employed at evaluation time.

4.2.3 Computational Setup

All the experiments of the main empirical results in Section 4.3 and Section 4.4 were run on a single GPU each, allocated from two separate Slurm-based internal clusters: the *CLuster pour l'Expérimentation et le Prototypage Scientifique* (CLEPS)[2], and the *Factory-IA* cluster[3]. We used the following GPU models from CLEPS: Nvidia V100, Nvidia RTX6000, Nvidia RTX8000, Nvidia GTX 1080Ti, Nvidia RTX2080Ti. And the following GPU models from the Factory-IA: Nvidia Quadro P5000, Nvidia H100.

4.2.4 Network Architectures

The `PreActResNet18` and `CNN-7` architectures used in our experiments are left unvaried with respect to the implementations from Jorge et al. [4]

[171]: Krizhevsky and Hinton (2009), 'Learning multiple layers of features from tiny images'

[170]: Netzer et al. (2011), 'Reading Digits in Natural Images with Unsupervised Feature Learning'

[124]: Paszke et al. (2019), 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

[76]: Ioffe and Szegedy (2015), 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'

and De Palma et al. [8], respectively.
The `CNN-5` architecture has the following structure:

1. convolutional layer with 64 3×3 filters, stride = 1 and padding = 1, followed by BatchNorm and a ReLU activation function;
2. convolutional layer with 64 4×4 filters, stride = 2 and padding = 1, then BatchNorm and ReLU;
3. convolutional layer with 128 4×4 filters, stride = 2 and padding = 1, then BatchNorm and ReLU;
4. linear layer with 512 neurons, then BatchNorm and ReLU;
5. linear layer with k (the number of classes) neurons.

All the networks trained using MTL-IBP, Exp-IBP and ForwAbs are initialized using the specialized technique from Shi et al. [98], which results in smaller IBP bounds at initialization. Except for the experiments from Section 4.1 and for those in Table 4.2, where the specialized initialization is employed in order to fairly compare IBP bounds, all the networks trained via pure adversarial training and ELLE (Section 4.3.3) are instead initialized using PyTorch's default initialization.

2: `https://paris-cluster-2019.gitlabpages.inria.fr`

3: `https://www.universite-paris-saclay.fr/plateforme-saclay-ia`

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

4.3 Preventing Catastrophic Overfitting

This section experimentally demonstrates that, on settings from the single-step adversarial training literature, Exp-IBP and ForwAbs can prevent CO when applied on top of single-step attacks.

4.3.1 FGSM

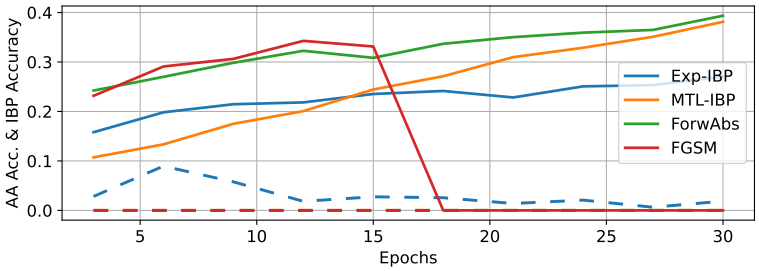
The seminal work from Wong et al. [6] established that FGSM, which is arguably the most vulnerable single-step attack, suffers from CO on CIFAR-10 from as early as $\epsilon = 8/255$. Therefore, assessing whether MTL-IBP, Exp-IBP and ForwAbs can be used to prevent CO when FGSM is used to generate the adversarial point x_{adv} is a crucial qualitative test. We consider two settings: $\epsilon = 8/255$, and the much harder $\epsilon = 24/255$. In line with the single-step adversarial training literature [4, 5, 7] and Section 4.1, we focus on `PreactResNet18`, trained via a cyclic learning rate schedule for 30 epochs. We run the three algorithms with a varying degree of regularization on their over-approximation tightness, plotting for each method the successful runs (*i.e.*, preventing CO) having the smallest α and λ values.
The goal of the experiment is to qualitatively discern whether certified training schemes can prevent catastrophic overfitting on FGSM. In order to determine this, we ran a series of Exp-IBP, MTL-IBP and ForwAbs experiments directly on the CIFAR-10 test set, and plotted the successful runs (in terms of preventing CO) with the smallest α or $\tilde{\lambda}$ values (among those tried), see caption of Figure 4.5 for details.

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

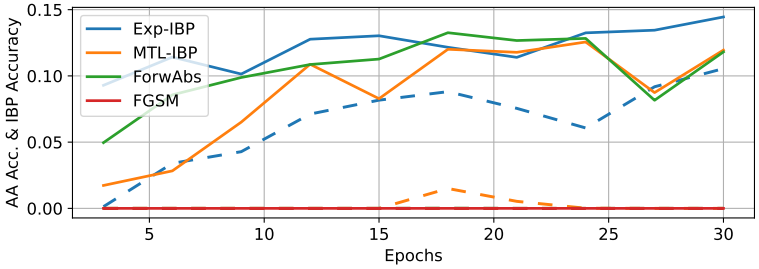
[5]: Andriushchenko and Flammarion (2020), 'Understanding and Improving Fast Adversarial Training'

SoftPlus While we focus on ReLU networks in all our experiments, as they are the most common in the catastrophic overfitting literature, and the certified training literature, we investigate the potential of certified training schemes to prevent CO on non-piecewise linear networks. In particular, we focus on a modified `PreActResNet18` architecture, for which each ReLU is replaced by a SoftPlus activation, relying on the training schedule from Figure 4.5.

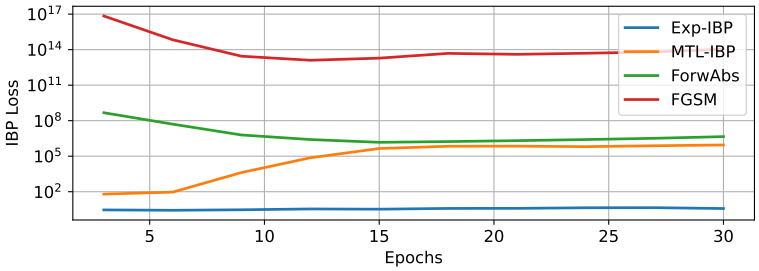
Definition 4.3.1 ***SoftPlus*** *is defined as* $\sigma(x) = \log(1 + e^x)$.



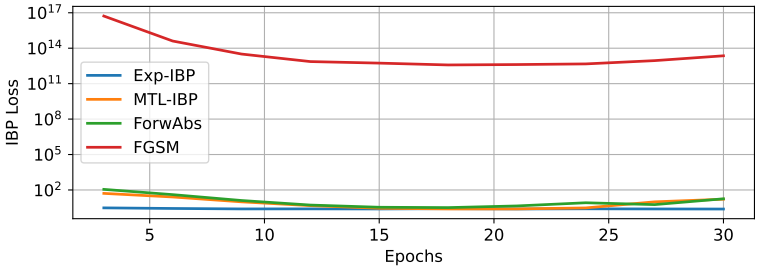
(a) AutoAttack (solid) and IBP (dashed) acc., $\epsilon = 8/255$.



(b) AutoAttack (solid) and IBP (dashed) acc., $\epsilon = 24/255$.



(c) IBP loss for $\epsilon = 8/255$.



(d) IBP loss for $\epsilon = 24/255$.

Figure 4.5: The use of certified training techniques on top of FGSM can prevent CO for `PreActResNet18` on the CIFAR-10 test set under perturbations of $\epsilon = 8/255$ and $\epsilon = 24/255$. Hyperparameters are as follows: on $\epsilon = 8/255$, $\alpha = 3 \times 10^{-2}$ for Exp-IBP, $\alpha = 10^{-8}$ for MTL-IBP, $\tilde{\lambda} = 10^{-18}$ for ForwAbs; on $\epsilon = 24/255$, $\alpha = 2.5 \times 10^{-2}$ for Exp-IBP, $\alpha = 10^{-7}$ for MTL-IBP, $\tilde{\lambda} = 2 \times 10^{-16}$ for ForwAbs.

Training schedule We use a short schedule popular in the literature [4–7]. The batch size is set to 128, and SGD with weight decay of 5×10^{-4} is used for the optimization. Crucially, no gradient clipping is employed, which (in addition to network depth and the lack of ramping up) we found to be a major factor behind the instability of pure IBP training (see Section 4.1.2). We train for 30 epochs with a cyclic learning rate linearly increasing from 0 to 0.2 during the first half of the training then decreasing back to 0.

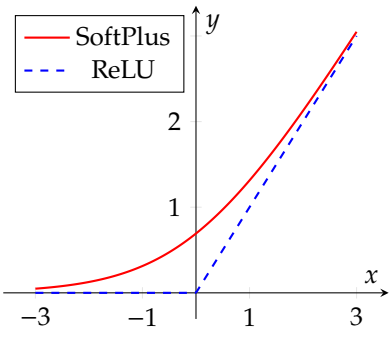
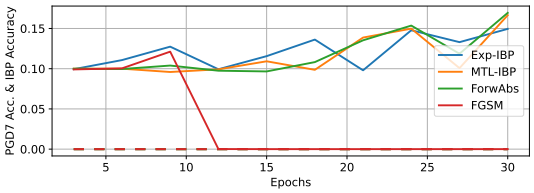
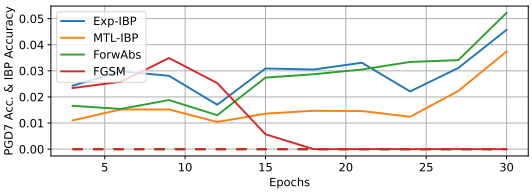


Figure 4.6: SoftPlus activation function, compared to ReLU.

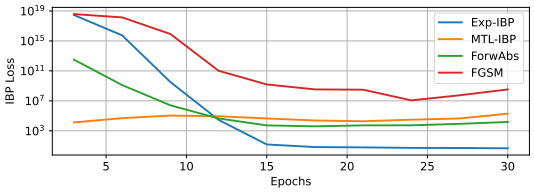
Owing to the monotonicity of SoftPlus, IBP bounds can be computed using the procedure outlined in Section 2.2.4 and using the Auto_LiRPA library, which we also employ for ReLU networks. We leave the ForwAbs implementation unvaried with respect to Section 4.1.3 Figure 4.7 provides results for CIFAR-10 and CIFAR-100 at $\epsilon = 24/255$, for which we show FGSM to be highly vulnerable to CO. Mirroring the experimental procedure for Figure 4.5, we demonstrate that MTL-IBP, Exp-IBP and ForwAbs can all prevent CO on this setting. Similarly to the ReLU experiments, this involves significantly decreasing the IBP loss compared to FGSM.



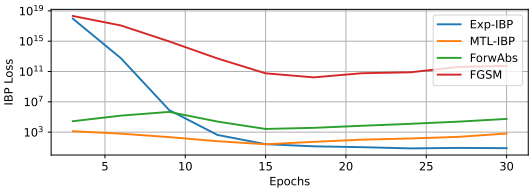
(a) PGD-7 (solid) and IBP (dashed) acc., CIFAR-10.



(b) PGD-7 (solid) and IBP (dashed) acc., CIFAR-100.



(c) IBP loss for CIFAR-10.



(d) IBP loss for CIFAR-100.

Figure 4.7: When applied on top of FGSM, certified training techniques can prevent CO beyond ReLU networks. Results with a modified `PreActResNet18` employing SoftPlus activations, for perturbations of $\epsilon = 24/255$ on the CIFAR-10 and CIFAR-100 test sets. The hyperparameters are as follows: on CIFAR-10, $\alpha = 5 \times 10^{-3}$ for Exp-IBP, $\alpha = 2 \times 10^{-11}$ for MTL-IBP, $\tilde{\lambda} = 10^{-20}$ for ForwAbs; on CIFAR-100, $\alpha = 1 \times 10^{-2}$ for Exp-IBP, $\alpha = 10^{-10}$ for MTL-IBP, $\tilde{\lambda} = 10^{-20}$ for ForwAbs. The training schedule for CIFAR-100 is the same as for CIFAR-10: 30 epochs with a cyclic learning rate linearly increasing from 0 to 0.2 during the first half of the training then decreasing back to 0.

4.3.2 N-FGSM

As explained in Section 2.2.2, the addition of noise on top of FGSM can mitigate CO at lower perturbation radii without any overhead. However, previous single-step adversarial training work [7, 92, 93] showed that even the state-of-the-art in noise-based single-step adversarial training, N-FGSM, becomes vulnerable when ϵ is larger. In order to complement the results of Figure 4.5, we now consider a more realistic setting where certified training schemes rely on adversarial points x_{adv} generated through N-FGSM, investigating the empirical robustness cost of preventing CO at large ϵ values from the literature [7]. As for the FGSM experiments, we use `PreactResNet18` with a 30-epoch cyclic training schedule. In these experiments we use empirical robustness to AutoAttack (AA) as the main performance criterion, and tune MTL-IBP, Exp-IBP and ForwAbs to maximize it on a holdout validation set, with $\epsilon = 24/255$ for CIFAR-10 and CIFAR-100, and with $\epsilon = 12/255$ for SVHN. As reported in Section 4.6.3, this tuning criterion will take a relatively heavy toll on standard performance. Nevertheless, Section 4.6 shows that, if desired, different tuning criteria may yield better trade-offs between clean and empirical robust accuracies. We then retrain on the full training set with the same chosen value for all the considered perturbation radii, reporting the test-set AA accuracy (and IBP accuracy when non-null). For this network, on CIFAR-100 and SVHN, we found the single-seed tuning employed in the rest of the paper to be a poor marker of final performance. Figure 4.9 reports values tuned to maximize mean validation PGD-50 accuracy on 5 seeds while preventing CO. Figure 4.8b shows an estimate of the training cost of each algorithm, computed on a single epoch on 80% of the CIFAR-10 training set: see Section 4.6.2 for further details.

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'

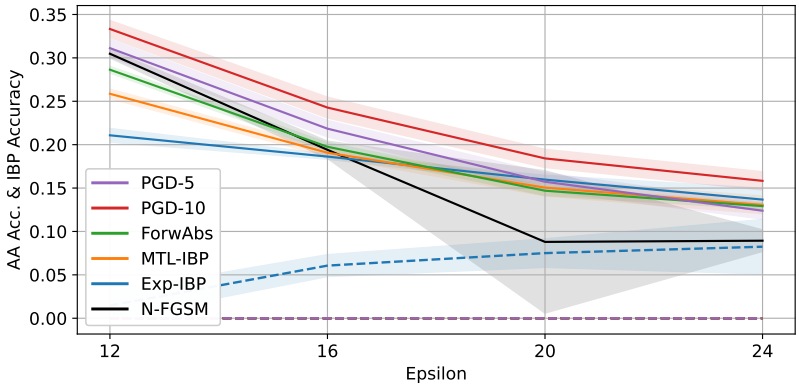
[92]: Lin et al. (2023), 'Eliminating catastrophic overfitting via abnormal adversarial examples regularization'

[93]: Lin et al. (2024), 'Layer-Aware Analysis of Catastrophic Overfitting: Revealing the Pseudo-Robust Shortcut Dependency'

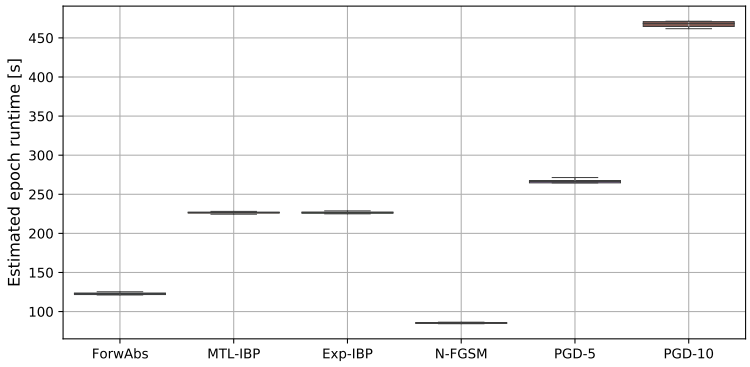
N-FGSM hyperparameters We use the same hyperparameters as Jorge et al. [4]: the uniform perturbation is sampled from $[-2\epsilon, 2\epsilon]$ for every setting except for SVHN with $\epsilon = 12/255$ where the perturbation is sampled from $[-3\epsilon, 3\epsilon]$. The step size is set to $\alpha = \epsilon$ for all settings.

Results Figure 4.8 shows that all the three considered algorithms can prevent CO for CIFAR-10, resulting in significantly larger robustness than N-FGSM on both $\epsilon = 20/255$ and $\epsilon = 24/255$. While certified training schemes pay a large price in empirical robustness for $\epsilon = 12/255$, they remarkably attain stronger AA robustness than PGD-5 for $\epsilon = 24/255$ with reduced runtime. Furthermore, for $\epsilon \geq 20/255$ Exp-IBP yields an IBP

accuracy comparable to the average AA accuracy of N-FGSM. As shown in Figure 4.9, CIFAR-100 and SVHN display instead worse trade-offs. While MTL-IBP, Exp-IBP and ForwAbs all prevent CO on CIFAR-100 for $\epsilon \geq 20/255$, as visible from the reduced confidence intervals, they do so at a large cost in average empirical robustness for $\epsilon < 24/255$. On the SVHN experiments, N-FGSM never suffers from CO. ForwAbs results in relatively small yet consistent performance improvements, while improvements from Exp-IBP are negligible. Remarkably, MTL-IBP fails to drive the IBP loss sufficiently low (cf. Figure 4.20c) and appears to be more vulnerable to CO than N-FGSM at $\epsilon = 12/255$. We ascribe this to the extreme sensitivity of MTL-IBP in the considered setup, linked to the qualitative properties of its loss on deeper networks described in Section 4.1.2 (see Figure 4.3). We exclude MTL-IBP from the rest of the empirical study owing to this failure case.



(a) AA (solid lines) and IBP (dashed) accuracies. Means over 5 repetitions and their 95% CIs.



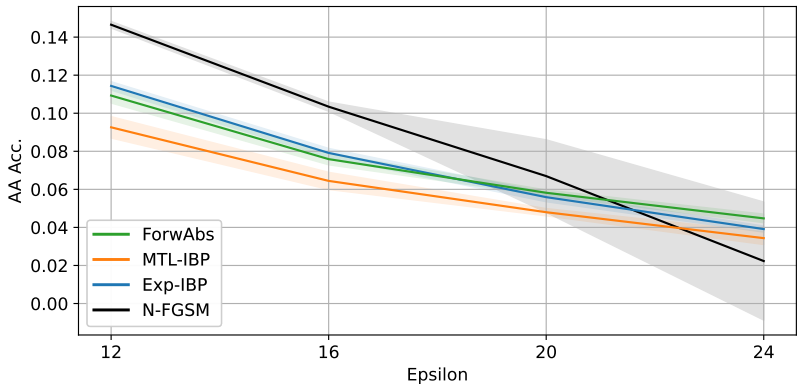
(b) Box plots (10 runs) for the training time of a single epoch, estimated on a 80% subset of the training set.

Figure 4.8: Certified training techniques can prevent CO for N-FGSM when training `PreactResNet18` on CIFAR-10, overcoming the robustness of PGD-5 for $\epsilon = 24/255$ while incurring less overhead. The training schedule is the 30 epochs schedule as in Figure 4.5.

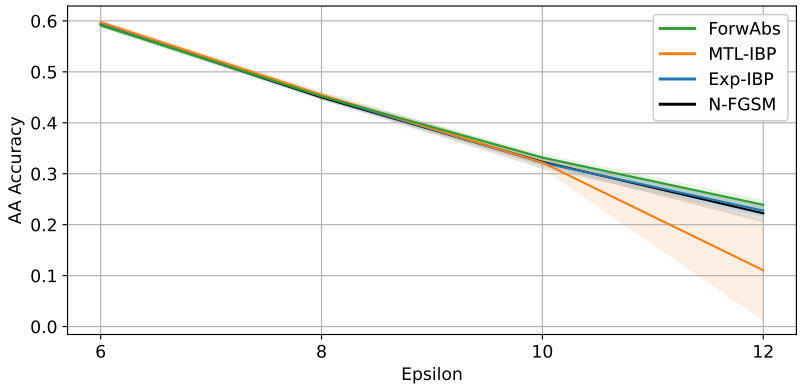
4.3.3 ELLE

The main objective of this work is to demonstrate that certified training techniques can be successfully employed toward empirical robustness, hence beyond their original design goal. As part of the provided evidence, Section 4.3.1 and Section 4.3.2 show that Exp-IBP and ForwAbs can prevent CO on settings common to the single-step adversarial training literature. In order to contextualize their performance for the task, we now provide a comparison of the performance of Exp-IBP, MTL-IBP and ForwAbs with ELLE-A [7], a state-of-the-art method designed to prevent catastrophic overfitting.

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'



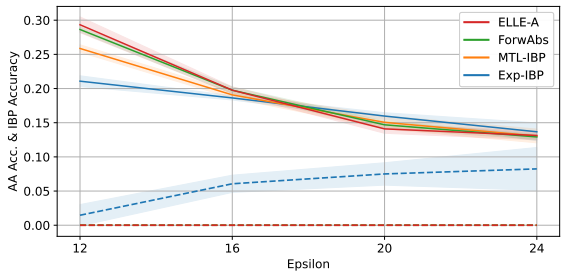
(a) CIFAR-100



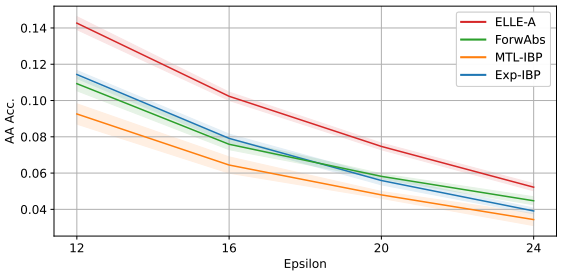
(b) SVHN

Figure 4.9: When training `PreactResNet18`, certified training techniques can prevent CO on CIFAR-100, albeit decreasing the average empirical robustness for perturbation radii they were not tuned for. N-FGSM does not display CO for SVHN on the same network: ForwAbs results nevertheless in minor average empirical robustness improvements, while MTL-IBP induces CO at $\epsilon = 12/255$ (means and 95% CIs over 5 runs). The CIFAR-100 is the same short schedule as CIFAR-10. On SVHN the training is done for 15 epochs, with a cyclic learning rate linearly increasing from 0 to 0.05 during 6 epochs, then decreasing back to 0 for the remaining 9 epochs. Furthermore, for SVHN only, the attack perturbation radius is ramped up from 0 to ϵ during the first 5 epochs, following Jorge et al. [4]

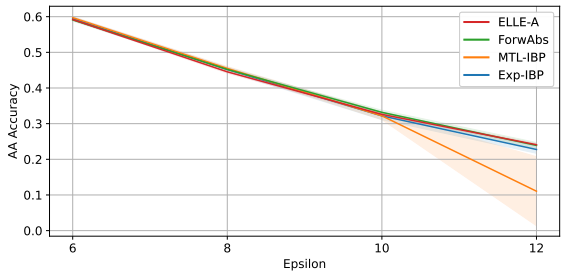
The experimental setup is the one from Figure 4.8 and Figure 4.9 (a `PreActResNet18` trained with a cyclic training schedule). As for the certified training techniques, we use N-FGSM as underlying adversarial attack. The runtime overhead of ELLE-A, which is slightly smaller than MTL-IBP and Exp-IBP, and significantly larger than ForwAbs on an Nvidia GTX 1080Ti GPU, is reported in Figure 4.10d. While the original work [7] reports less overhead for ELLE-A, which requires 3 batched forward passes to compute its regularization term, we found its overhead to be heavily dependent on the GPU model, with newer models leading to a faster execution of the batched forward pass (the original work relies on an Nvidia A100 SXM4 GPU). We tuned the ELLE-A hyper-parameter λ_{ELLE}, which controls the amount of regularization imposed on a notion of local linearity proposed by the authors, consistently with the way MTL-IBP, Exp-IBP and ForwAbs were tuned for Figure 4.8 and Figure 4.9. The tuning resulted in the following values: $\lambda_{\text{ELLE}} = 6000$ for CIFAR-10 (which maximizes both the validation PGD-50 and AA accuracy), $\lambda_{\text{ELLE}} = 3000$ for CIFAR-100, and $\lambda_{\text{ELLE}} = 1000$ for SVHN.
As reported in Figure 4.10, ELLE-A prevents CO across all settings. Remarkably, on CIFAR-10, certified training techniques match or outperform ELLE-A in AA accuracy for $\epsilon \geq 20/255$, with Exp-IBP outperforming ELLE-A while also producing non-negligible certified robustness via IBP. We believe this suggests that verifiability and empirical robustness are not conflicting objectives in this setup. Furthermore, ForwAbs performs at least competitively with ELLE-A on all the considered epsilons while reducing its overhead. The situation is drastically different on the harder CIFAR-100, where ELLE-A markedly outperforms certified training schemes. We speculate the employed network may lack the



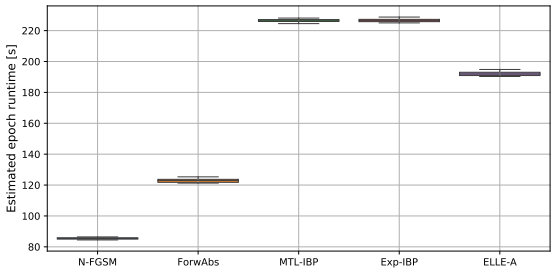
(a) CIFAR-10, AA (solid lines) and IBP accuracies (dashed).



(b) CIFAR-100.



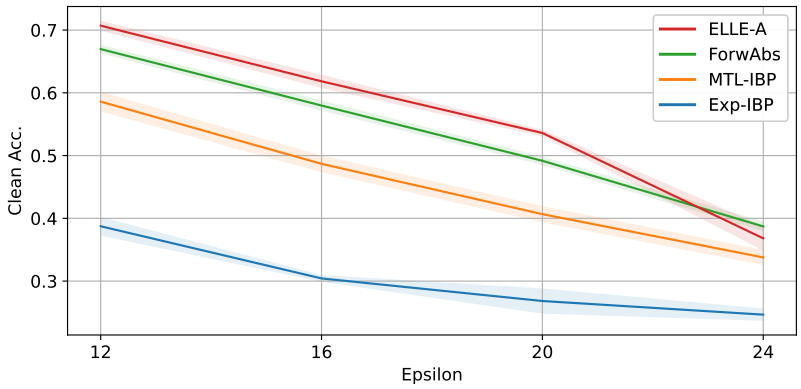
(c) SVHN.



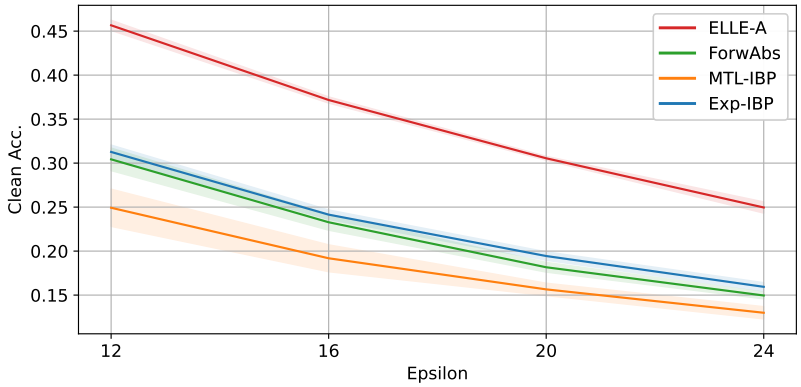
(d) Box plots (10 runs) for the CIFAR-10 training time of a single epoch, estimated on a 80% subset of the training set.

Figure 4.10: Comparison of certified training techniques with ELLE-A [7], a the state-of-the-art regularizer for singe-step adversarial training. Setup from Figure 4.8 and Figure 4.9. We report means over 5 repetitions and their 95% confidence intervals.

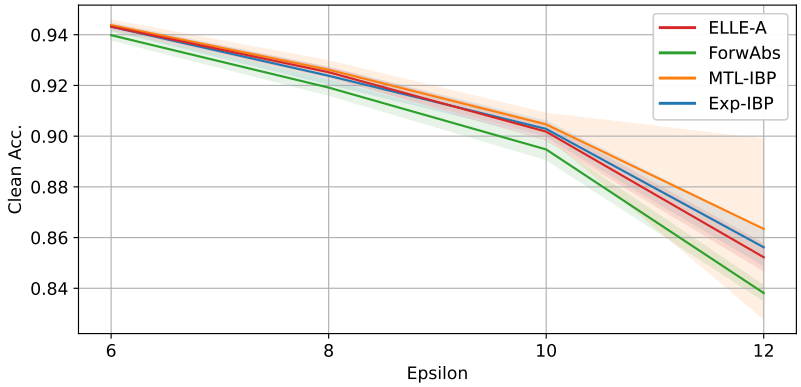
capacity to sustain tight over-approximations while preserving empirical robustness. On SVHN, except MTL-IBP which fails owing to the sensitivity of its loss as discussed in Section 4.3.2, all the algorithms attain similar performance profiles, with the most robust algorithm depending on the perturbation radius. Figure 4.11 reports the corresponding clean accuracies: on average ELLE-A has less impact on clean accuracy than certified training schemes, except for SVHN, where expressive losses display larger standard performance, and on CIFAR-10 for $\epsilon = 24/255$, where ForwAbs does. In summary, we argue that, on easier datasets such as CIFAR-10, certified training techniques such as Exp-IBP and ForwAbs should be considered as strong baselines for CO prevention by the single-step adversarial training community.



(a) CIFAR10.



(b) CIFAR-100.

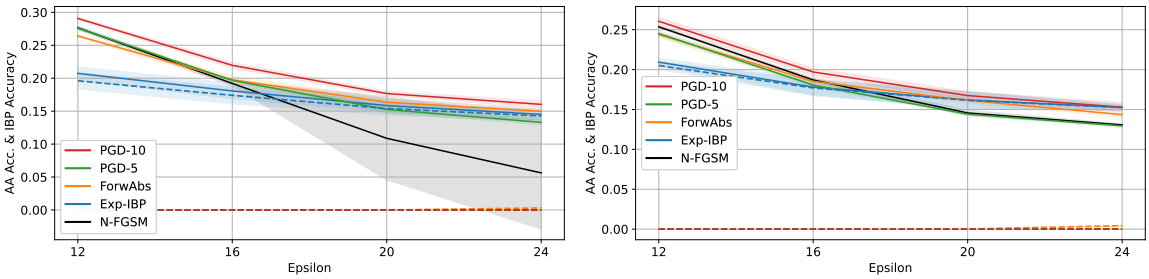


(c) SVHN.

Figure 4.11: Clean accuracies for the experiments reported in Figure 4.10

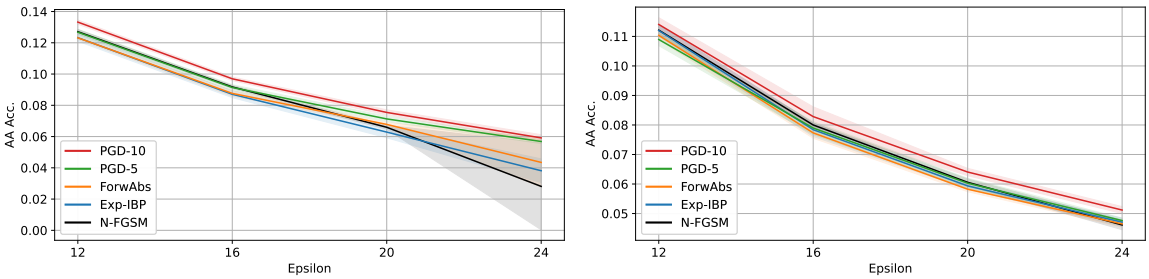
4.4 Bridging the Gap to Multi-Step Adversarial Training

We here demonstrate that the gap between certified training techniques based on single-step attacks and multi-step baselines can be reduced by training on shallower networks and with longer schedules, in some settings outperforming PGD-10 in empirical robustness without employing an adversarial training component.



(a) `CNN-7`. **(b)** `CNN-5`.

Figure 4.12: When training `CNN-7` and `CNN-5` on CIFAR-10, ForwAbs and Exp-IBP prevent CO while displaying stronger empirical robustness than PGD-5 for $\epsilon \geq 20/255$, and matching PGD-10 for `CNN-5` at $\epsilon = 24/255$. AutoAttack (solid lines) and IBP (dashed) accuracies are reported (means and 95% CIs for 5 runs).
As seen in Table 4.1, `CNN-7` features an IBP loss almost two orders of magnitude larger than `CNN-5`, explaining the qualitative differences in this figure.



(a) `CNN-7`. **(b)** `CNN-5`.

Figure 4.13: When training `CNN-7` and `CNN-5` for CIFAR-100, ForwAbs and Exp-IBP display better performance trade-offs than on the deeper `PreActResNet18` but still fail to improve on multi-step attacks. AutoAttack (solid lines) and IBP (dashed) accuracies are reported (means and 95% CIs for 5 runs).

4.4.1 Cyclic training schedule

Experimental setting We replicate the cyclic-schedule experiments from Figure 4.8 and Figure 4.9 on two shallower networks without skip connections. As in Figure 4.8 and Figure 4.9, x_{adv} for certified training methods is generated via N-FGSM. First, we consider `CNN-7`, a 7-layer convolutional network from the certified training literature [8, 98, 104, 105]. Then, to study the impact of network depth, we also provide results on a 5-layer version, named `CNN-5`.
We tune Exp-IBP and ForwAbs to maximize AA accuracy for $\epsilon = 24/255$ on a validation set for each dataset and network, using the tuned value for all considered ϵ. As discussed in Section 4.3 and reported in Section 4.6.3 this negatively impact the networks standard performance: while this is not the focus of these experiments, better robustness-accuracy trade-offs may be obtained using different tuning criteria (see Section 4.6.1).

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[104]: Mao et al. (2023), 'TAPS: Connecting Certified and Adversarial Training'

[105]: Müller et al. (2023), 'Certified Training: Small Boxes are All You Need'

Results Figure 4.12 demonstrates that both ForwAbs and Exp-IBP outperform PGD-5 in empirical robustness for $\epsilon \geq 20/255$ on both `CNN` models, with ForwAbs preserving AA accuracy at lower epsilons, and Exp-IBP attaining significant IBP accuracy (at least as large as PGD-5's AA accuracy for $\epsilon \geq 20/255$). The gap to PGD-10 is also significantly reduced compared to the `PreActResNet18` experiments, in spite of the significant CO shown by N-FGSM, the attack employed within both Exp-IBP and ForwAbs. Furthermore, as shown in Section 4.6.2, both Exp-IBP and ForwAbs reduce runtime compared to PGD-5 and PGD-10. Remarkably, on `CNN-5`, as shown in Figure 4.12b, Exp-IBP matches the performance of PGD-10 for $\epsilon = 24/255$, significantly outperforming N-FGSM, which

does not suffer from CO in this setup. In order to prevent ForwAbs from under-performing on `CNN-5`, we found it crucial to include sufficiently low λ values in the tuning grid.
Finally, Figure 4.13 shows that, while the overall performance trade-offs are greatly improved compared to `PreActResNet18`, Exp-IBP and ForwAbs fail to improve on multi-step attacks on CIFAR-100 also on the `CNN` architectures. We ascribe this to the larger number of classes, which produce a less favorable trade-off between empirical and certified robustness. Table 4.1 shows that the `PreActResNet18` IBP loss at initialization is more than a factor 10^{10} larger than on the `CNN` architectures, providing an intuitive explanation for the different performance profiles.

4.4.2 Long training schedule

Experimental setting We now investigate whether certified training schemes can be more beneficial under longer training schedules. The long training schedule used for the experiments in Table 4.2 mirror a setup from Shi et al. [98], widely adapted in the certified training literature [8, 104, 105]. Training is carried out for 160 epochs using the Adam optimizer with a learning rate of 5×10^{-4}, decayed twice by a factor of 0.2 at epochs 120 and 140. Gradient clipping is employed, with the maximal ℓ_2 norm of gradients equal to 10. Training starts with an epoch of clean training ("warm-up"). During epochs 1 to 81, the perturbation radius (regardless of the computation it is employed for) is increased from 0 to its target value using the `SmoothedScheduler` from Auto_LiRPA Xu et al. [46]. For Exp-IBP and IBP, we employ a specialized regularizer [98] for the IBP bounds for the first 81 epochs (with coefficient 0.5 as done by the original authors on CIFAR-10), as commonly done in previous work [8, 104, 105].
In order to show the benefits of tuning for each ϵ, the hyperparameters of Exp-IBP and ForwAbs (α and λ, respectively) are tuned to maximize validation AA accuracy individually on each setup considered in Table 4.2, with the exception of $\epsilon = 16/255$ CIFAR-10, which re-uses the values from $\epsilon = 24/255$ to reduce computational overhead. As in Figure 4.8a to Figure 4.13, N-FGSM is used to generate $\boldsymbol{x}_{\text{adv}}$ for Exp-IBP and ForwAbs, except for "Exp-IBP PGD-5", which relies on PGD-5 instead. Noticing the beneficial effect of large α values, we also include pure IBP ($\alpha = 1$), whose loss is attack-free, in the comparison.

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[105]: Müller et al. (2023), 'Certified Training: Small Boxes are All You Need'

	CIFAR-10 $\epsilon = 8/255$		CIFAR-10 $\epsilon = 16/255$		CIFAR-10 $\epsilon = 24/255$		CIFAR-100 $\epsilon = 24/255$	
Method	AA acc. [%]	IBP acc. [%]	AA acc. [%]	IBP acc. [%]	AA acc. [%]	IBP acc. [%]	AA acc. [%]	IBP acc. [%]
Exp-IBP	*38.44 ± 0.25*	0.00	*20.72 ± 1.54*	20.55 ± 1.45	*14.79 ± 1.61*	14.74 ± 1.59	3.25 ± 0.44	3.14 ± 0.41
Exp-IBP PGD-5					*16.45 ± 1.32*	16.34 ± 1.29		
IBP	29.43 ± 0.96	**29.02 ± 0.96**	**21.72 ± 0.68**	**21.62 ± 0.68**	**16.73 ± 0.43**	**16.56 ± 0.42**	3.78 ± 0.16	**3.67 ± 0.16**
ForwAbs	*37.92 ± 0.08*	0.00	17.61 ± 0.38	0.00	*14.39 ± 0.16*	0.46 ± 0.17	4.31 ± 0.05	0.00
PGD-10	**40.14 ± 0.65**	0.00	21.13 ± 0.60	0.00	*15.14 ± 0.46*	0.00	**6.15 ± 0.16**	0.00
PGD-5	37.32 ± 0.71	0.00	18.19 ± 0.32	0.00	11.47 ± 0.29	0.00	5.52 ± 0.18	0.00
N-FGSM	37.76 ± 0.30	0.00	7.13 ± 11.43	0.00	0.23 ± 0.64	0.00	0.02 ± 0.06	0.00

Table 4.2: When training `CNN-7` with the long training schedule, Exp-IBP consistently improves on the average empirical robustness of PGD-5 on CIFAR-10, with IBP outperforming PGD-10 for $\epsilon \geq 16/255$. Multi-step adversarial training displays the best performance on CIFAR-100. Bold entries indicate the best AA or IBP accuracy for each setting. Italics denote AA accuracy improvements on PGD-5. Means and 95% CIs for 5 runs are reported.

Results Table 4.2 shows that Exp-IBP outperforms PGD-5 on all considered CIFAR-10 setups. While empirical robustness is maximized when the IBP accuracy is large on $\epsilon = 24/255$, it is not the case for $\epsilon = 8/255$. Remarkably, pure IBP training outperforms all other methods for $\epsilon \geq 16/255$ on CIFAR-10. Exp-IBP PGD-5 demonstrates that on CIFAR-10 certified training techniques can also improve robustness when applied on top of multi-step attacks: this appears to be useful to improve performance when single-step attacks systematically fail. While ForwAbs can significantly

Table 4.3: Effect of model architecture on AutoAttack accuracy on CIFAR-10 with $\epsilon = 24/255$. Results from Figure 4.8a, Figure 4.12a, and Figure 4.12b (means and standard deviations for 5 runs).

Architecture	Exp-IBP AA acc. [%]	ForwAbs AA acc. [%]	PGD-5 AA acc. [%]	PGD-10 AA acc. [%]
`PreActResNet18`	13.67 ± 1.30	12.92 ± 0.34	12.39 ± 0.96	15.83 ± 1.05
`CNN-7`	14.50 ± 0.81	14.98 ± 0.37	13.29 ± 0.37	16.03 ± 0.52
`CNN-5`	15.26 ± 0.35	14.36 ± 0.32	12.94 ± 0.23	15.28 ± 0.58

boost the empirical robustness of N-FGSM, it manages to outperform PGD-5 only on CIFAR-10 with $\epsilon = 24/255$. Finally, the long schedule does not benefit certified training schemes on CIFAR-100, where their relative performance compared to multi-step attacks remains unvaried compared to the cyclic schedule, and for which we did not find the use of PGD-5 to generate the attacks for Exp-IBP to be beneficial.

4.4.3 Effect of Model Architecture on Method Performance

Table 4.3 presents an analysis of the effect of model size on the performance of both multi-step adversarial training and certified training techniques, grouping in table form the results from Figure 4.8a, Figure 4.12a, and Figure 4.12b on CIFAR-10 with $\epsilon = 24/255$. Differently from results of the previous sections, where the focus was on relative performance across methods, here we focus on the impact of model architecture on the performance of each method. All considered algorithms attain larger empirical robustness on `CNN-7` compared to `PreActResNet18`. Remarkably, and differently from all the other techniques, the AutoAttack accuracy of Exp-IBP is maximized on `CNN-5`, suggesting that smaller networks may be particularly beneficial to expressive losses under shorter training schedules.

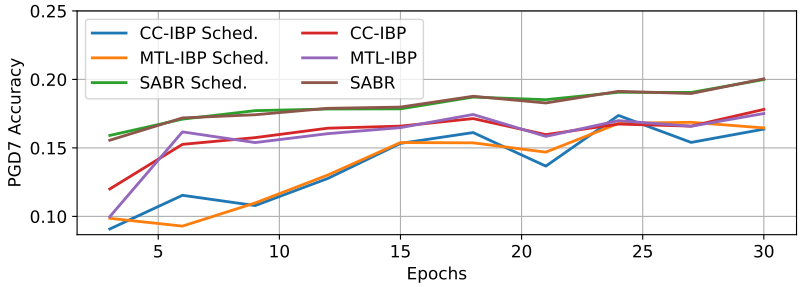
4.5 Hyperparameters and scheduling

Table 4.4 summarizes the hyperparameters employed in all experiments.
All the MTL-IBP and ForwAbs experiments with the short schedules gradually increase both the method coefficient (respectively α and λ) and the perturbation radius used to compute the IBP bounds (or their proxy $\tilde{\delta}^n$) from 0 to their target value. We conduct an ablation study on MTL-IBP in the context of the experiment from Figure 4.2.
Figure 4.14 studies the effect of scheduling (gradually increasing from 0 to their target value in 25 epochs) both α and the perturbation radius ϵ employed to compute the IBP lower bounds $\boldsymbol{l}_{f_\theta}^{B(\boldsymbol{x},\epsilon),y}$. Excluding trivial outcomes, such as networks systematically outputting the same class, for CC-IBP, MTL-IBP and SABR, we were unable to reach validation IBP loss values below 44 without scheduling. The scheduling allows the use of larger α values, resulting in lower IBP loss value for CC-IBP, MTL-IBP

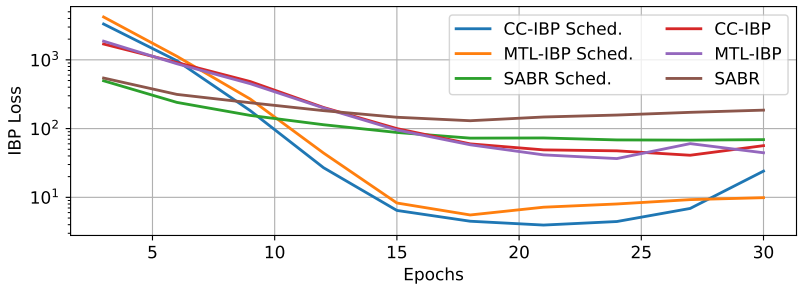
Table 4.4: Exp-IBP, MTL-IBP and ForwAbs coefficients for figures 4.8 to 4.13, figure 4.19, figure 4.20, and Table 4.2.

Experiment	Exp-IBP α	MTL-IBP α	ForwAbs $\tilde{\lambda}$
Figures 4.8a, 4.19a and 4.20a	2×10^{-2}	1×10^{-8}	1×10^{-24}
Figures 4.9a, 4.19g and 4.20b	5×10^{-3}	1×10^{-8}	1×10^{-18}
Figures 4.9b, 4.19c and 4.20c	2×10^{-3}	2.75×10^{-15}	1×10^{-23}
Figures 4.12b, 4.19e and 4.20e	4×10^{-1}		1×10^{-8}
Figures 4.13b, 4.19e and 4.20g	2×10^{-3}		1×10^{-10}
Figures 4.12a, 4.19d and 4.20d	2×10^{-1}		1×10^{-10}
Figures 4.13a, 4.19f and 4.20f	2×10^{-3}		1×10^{-12}
Table 4.2, CIFAR-10, $\epsilon = 8/255$	5×10^{-3}		1×10^{-12}
Table 4.2, CIFAR-10, $\epsilon = 24/255$ and $\epsilon = 16/255$	7.5×10^{-1}		1×10^{-8}
Table 4.2, CIFAR-100, $\epsilon = 24/255$	7.5×10^{-1}		1×10^{-8}

and SABR: respectively from 56.47 to 24.06, from 44.50 to 9.89, and from 185.51 to 69.09. These improvements come with a slight decrease in robustness to PGD-7 (a PGD attack with 7 steps). Nevertheless, CC-IBP, MTL-IBP and SABR are unable to attain non-negligible certified accuracy via IBP in this context. In view of these results, we employ the same scheduling for ForwAbs.



(a) Empirical accuracy under PGD7 attacks.



(b) IBP loss.

Figure 4.14: Effect of scheduling the bounding perturbation radius and α on the IBP certified robustness attained by CC-IBP, MTL-IBP and SABR on the `PreActResNet18` training setup from Jorge et al. [4]. Validation results on CIFAR-10 under perturbations of $\epsilon = 8/255$. Hyperparameters are: $\alpha = 0.1$ for Exp-IBP, $\alpha = 10^{-6}$ for CC-IBP and MTL-IBP with scheduling, $\alpha = 10^{-15}$ for CC-IBP and MTL-IBP without scheduling, $\alpha = 10^{-4}$ for SABR with scheduling and $\alpha = 10^{-9}$ for SABR without scheduling. They are chosen to reduce the IBP loss as much as possible on the validation set.
[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

The increase happens over the first 25 epochs for CIFAR-10 and CIFAR-100, and over the first 12 epochs for SVHN. In particular, the value is increased exponentially for the first 25% of the above epochs and linearly for the rest [98], relying on the `SmoothedScheduler` from Auto_LiRPA [46]. The radius used to compute the attack is kept consistent with the adversarial training literature (that is, constant in all cases except for the first 5 epochs on SVHN).

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

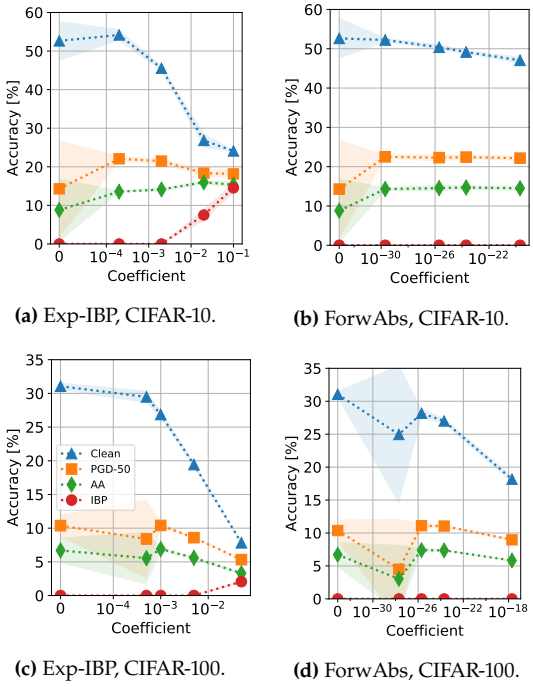
[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

4.6 Sensitivity Analysis and Performance Trade-Offs

We discuss in this section the main limitations of the proposed methods: the sensitivity to the tuned coefficients associated with the losses, and the trade-offs between empirical robustness and standard performance.

4.6.1 Sensitivity Analysis

This section presents a study of the test-set behavior of Exp-IBP and ForwAbs for varying values of their coefficients (respectively α and λ) when training `PreActResNet18` and `CNN-7` with the cyclic schedule. Figures 4.8a, 4.9a, 4.12a and 4.13a display results tuned to maximize AA robustness on $\epsilon = 24/255$. We here focus on $\epsilon = 20/255$ to potentially showcase different performance profiles, either in terms of maximizing empirical robustness or in terms of trade-offs with standard accuracy.
For `PreActResNet18`, figure 4.15 shows that, on CIFAR-10, CO can be mitigated without incurring a significant cost in standard performance.



(a) Exp-IBP, CIFAR-10. **(b)** ForwAbs, CIFAR-10.

(c) Exp-IBP, CIFAR-100. **(d)** ForwAbs, CIFAR-100.

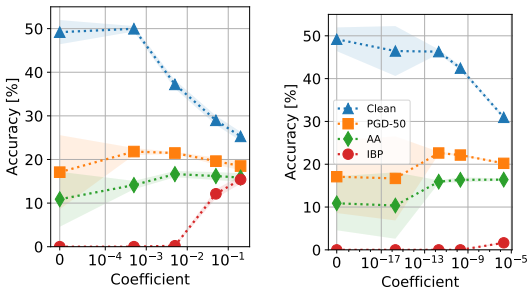
Figure 4.15: Sensitivity of Exp-IBP and ForwAbs to their respective coefficients, α and λ, when training `PreActResNet18` for ℓ_∞ perturbations of $\epsilon = 20/255$. Plot 4.15c displays the legend for all sub-figures, which log standard accuracy (Clean), empirical robust accuracies to PGD-50 and AutoAttack (AA), and IBP verified robust accuracy on the standard test sets.

On the other hand, maximizing empirical robustness alone, especially if with respect to AutoAttack accuracy, will result in a larger standard performance drop: this is further highlighted by the clean accuracies relative to figures 4.8 to 4.13 and Table 4.2, which are reported in Section 4.6.3. Stronger regularization is required on CIFAR-100, where low coefficient values appear to be detrimental to robustness compared to N-FGSM (corresponding to $\alpha = 0$ or $\lambda = 0$). Nevertheless, differently from the results showed in Figure 4.9a, both Exp-IBP and ForwAbs can improve on the average empirical robustness of N-FGSM, highlighting their versatility and the potential advantages of per-instance tuning. Figure 4.16 shows similar trends for CIFAR-10 on `CNN-7`, where however both methods attain a larger AutoAttack accuracy, and at a smaller cost in standard performance.

4.6.2 Training Overhead

Figure 4.8b shows the per-epoch training overhead of MTL-IBP, Exp-IBP, ForwAbs, PGD-5 and PGD-10 with respect to N-FGSM (which is used to compute the attack for MTL-IBP, Exp-IBP, ForwAbs) when training a `PreActResNet18` on 80% of the CIFAR-10 training set using the cyclic schedule, complementing the information provided in Figure 4.4. As described in Section 4.1.3, ForwAbs has minimal overhead with respect to N-FGSM. On the other hand, MTL-IBP and Exp-IBP display a runtime slightly smaller than PGD-5. As expected, PGD-10 training requires almost twice the time as PGD-5, and more than half the time as MTL-IBP and Exp-IBP. Figure 4.17, instead, plots the respective training overheads when training `CNN-7` and `CNN-5` using the cyclic schedule. The relative runtimes across methods remain similar to `PreActResNet18`, with the runtime of each algorithm markedly smaller on the `CNN` models, and for the 5-layer network. Differently from the experiments in Section 4.3 and Section 4.4, all these runtime measurements were carried out on the same machine under constant load. [4] Figure 4.18 explicitly plots trade-offs between runtime and empirical robustness. In particular, figures 4.18a and 4.18c respectively plot the runtimes from

4: All timing measurements were carried out on an Nvidia GTX 1080Ti GPU, using 6 cores of an Intel Skylake Xeon 5118 CPU.



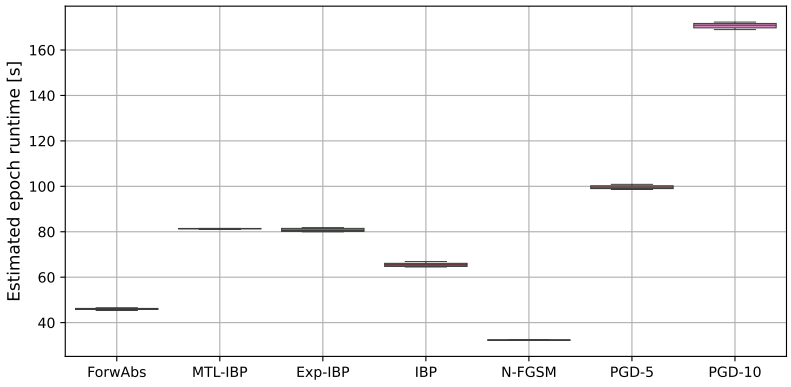
(a) Exp-IBP, CIFAR-10.

(b) ForwAbs, CIFAR-10.

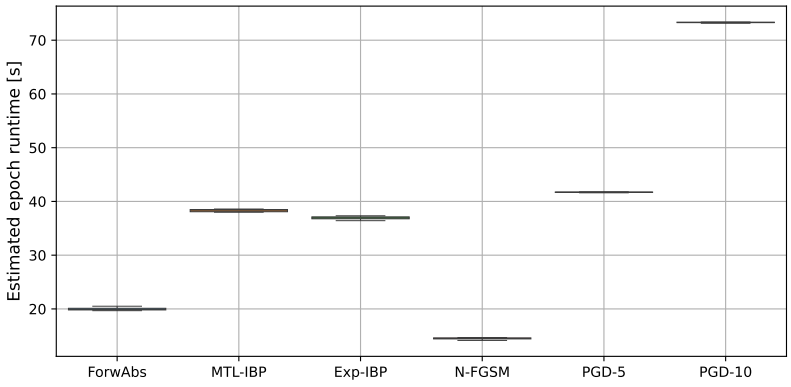
(c) Exp-IBP, CIFAR-100.

(d) ForwAbs, CIFAR-100.

Figure 4.16: Sensitivity of Exp-IBP and ForwAbs to their respective coefficients, α and λ, on `CNN-7` for ℓ_∞ perturbations of $\epsilon = 20/255$ using the cyclic training schedule. Plot 4.16b displays the legend for all sub-figures.



(a) `CNN-7`.



(b) `CNN-5`.

Figure 4.17: Box plots (10 repetitions) for the CIFAR-10 training time of a single epoch (using a 80% subset of the training set) on `CNN-7` and `CNN-5`.

figures 4.17a and 4.8b against the AA accuracies reported for CIFAR-10 at $\epsilon = 24/255$ within figures 4.8a and 4.12a. Differently from the cyclic schedule, the long schedule (160 epochs) used for the experiments of Table 4.2 does not have a constant training cost throughout the epochs (see Section

4.5): all methods start with a warm-up epoch that requires additionally evaluating the network on clean inputs, and methods using IBP bounds (Exp-IBP and IBP) employ the regularizer from Shi et al. [98] to control the bounds in earlier epochs, increasing their overhead until epoch 81. Figure 4.18b plots the AA accuracies for CIFAR-10 at $\epsilon = 24/255$ from Table 4.2 against the per-epoch runtimes from Figure 4.17a for N-FGSM, PGD-5, PGD-10 and ForwAbs (hence providing an estimate of long-schedule runtime after warm-up, on 80% of the CIFAR-10 training set), and against the per-epoch runtimes (also computed on 80% of the training set, after warm-up) of Exp-IBP and IBP when the bounds regularizer is active. Across the three setups, both ForwAbs and Exp-IBP consistently outperform PGD-5, with the former incurring a relatively small overhead compared to N-FGSM. On the long schedule, IBP outperforms all the other algorithms despite its low runtime, including the stronger multi-step baseline PGD-10.

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

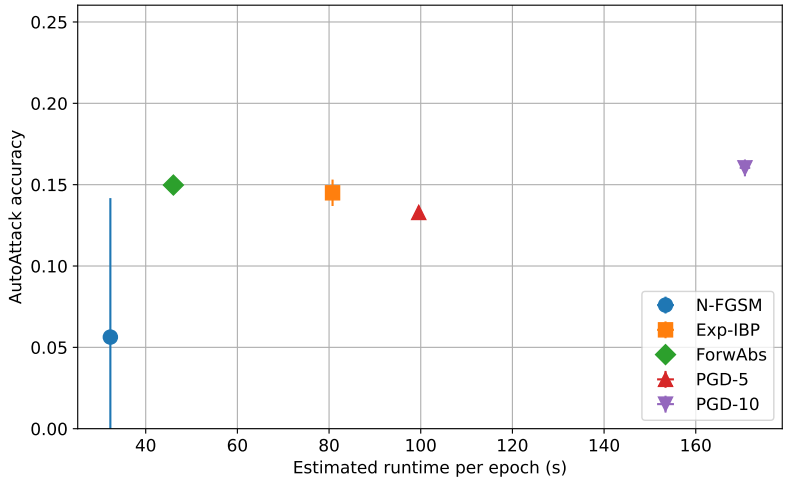
4.6.3 Clean Accuracies and IBP Losses

Table 4.5: Clean accuracies and IBP losses for the experiments in Table 4.2.

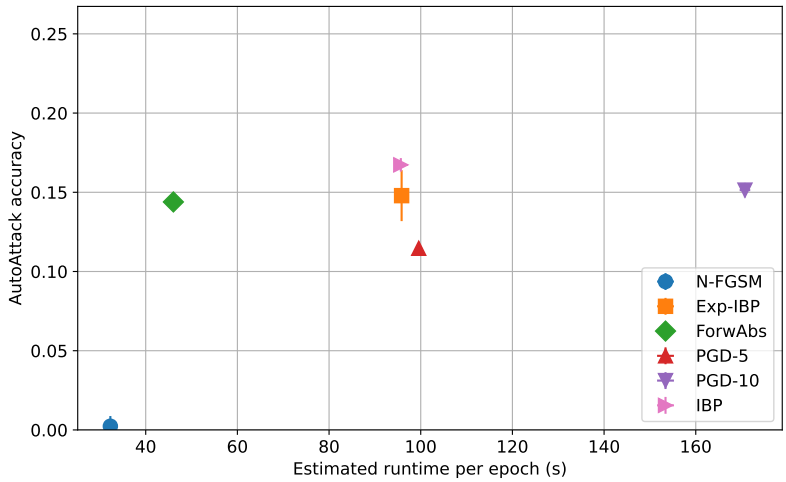
Dataset	ϵ	Method	Clean acc. [%]	IBP loss
CIFAR-10	$\frac{8}{255}$	Exp-IBP	76.30 ± 0.43	(4.90 ± 0.55) × 10^4
		IBP	39.37 ± 1.51	1.92 ± 0.01
		ForwAbs	74.26 ± 0.28	(2.05 ± 0.11) × 10^4
		PGD-10	77.07 ± 0.31	(9.79 ± 0.85) × 10^5
		PGD-5	79.81 ± 0.38	(1.72 ± 0.11) × 10^6
		N-FGSM	77.28 ± 0.26	(1.91 ± 0.14) × 10^6
	$\frac{16}{255}$	Exp-IBP	31.30 ± 2.13	2.15 ± 0.02
		IBP	31.65 ± 0.55	2.13
		ForwAbs	46.54 ± 0.31	(7.72 ± 0.37) × 10^1
		PGD-10	58.44 ± 0.24	(3.57 ± 0.33) × 10^5
		PGD-5	64.07 ± 0.14	(1.01 ± 0.13) × 10^6
		N-FGSM	62.38 ± 9.16	(2.93 ± 2.66) × 10^6
	$\frac{24}{255}$	Exp-IBP	23.03 ± 4.97	2.25 ± 0.02
		Exp-IBP PGD-5	27.32 ± 2.94	2.23 ± 0.02
		IBP	23.51 ± 2.63	2.22 ± 0.01
		ForwAbs	30.15 ± 0.37	8.73 ± 0.53
		PGD-10	39.99 ± 0.60	(2.60 ± 0.42) × 10^4
		PGD-5	49.31 ± 0.37	(2.13 ± 0.22) × 10^5
		N-FGSM	57.46 ± 9.60	(3.31 ± 1.55) × 10^6
CIFAR-100	$\frac{24}{255}$	Exp-IBP	6.85 ± 1.29	4.49 ± 0.02
		IBP	7.29 ± 0.61	4.45 ± 0.01
		ForwAbs	16.85 ± 0.67	(5.79 ± 0.36) × 10^1
		PGD-10	23.56 ± 0.61	(2.19 ± 0.19) × 10^5
		PGD-5	28.53 ± 0.46	(6.12 ± 0.32) × 10^5
		N-FGSM	29.07 ± 3.01	(6.50 ± 1.92) × 10^6

We here provide omitted clean accuracies and IBP losses for the experiments for Figure 4.8 to Figure 4.13 and Table 4.2. Figure 4.19 provides the results for the cyclic training schedule on `PreActResNet18`, `CNN-7` and `CNN-5`. It is clear that the positive effects associated to the certified training schemes (the mitigation or prevention of catastrophic overfitting in Section 4.3 and the empirical robustness improvements detailed in Section 4.4) come at a cost in clean accuracy, which is a limitation of these methods and of enforcing low IBP loss. Generally speaking, as visible in Figure 4.20, Exp-IBP attains significantly better trade-offs between

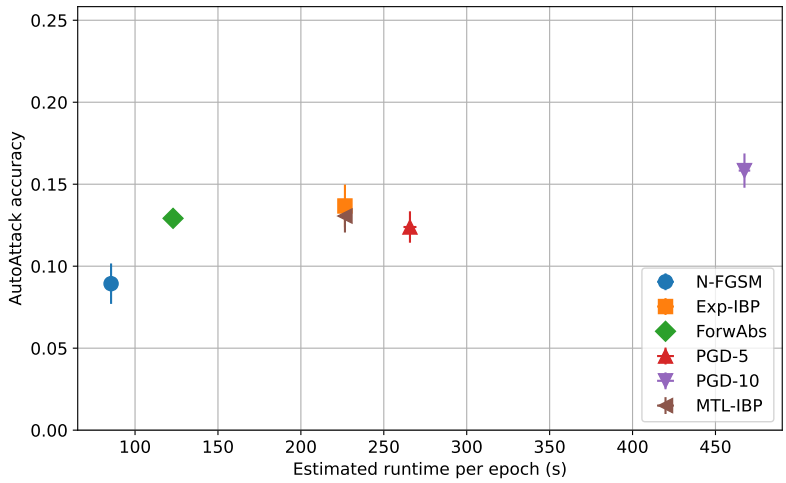
clean accuracy and IBP loss, especially when compared to MTL-IBP. As a result, Exp-IBP is to be preferred if IBP accuracy is the primary metric. Table 4.5 presents the clean accuracies and IBP losses pertaining to the long schedule experiment from Table 4.2, additionally presenting a comparison between the IBP losses of the certified training schemes tuned for empirical robustness and the adversarial training baselines whose IBP loss was shown in Figure 4.1. In all cases, certified training schemes decrease the IBP loss compared to the adversarial training baselines at a cost in standard accuracy, with a larger cost associated to a greater reduction in IBP loss. We conclude by pointing out that an alternative tuning criterion (considering trade-offs between standard accuracy and empirical robustness) could result in more favorable trade-offs. This was not the focus of the presented experiments.



(a) `CNN-7`, CIFAR-10, $\epsilon = 24/255$, setup of Figure 4.12a.

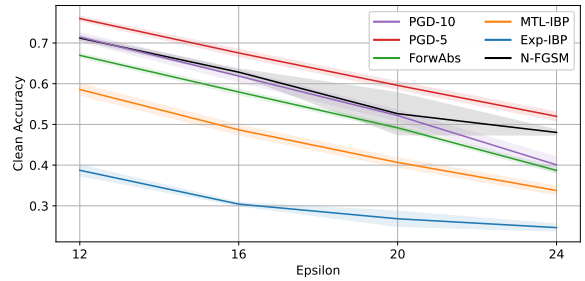


(b) `CNN-7`, CIFAR-10, $\epsilon = 24/255$, setup of Table 4.2

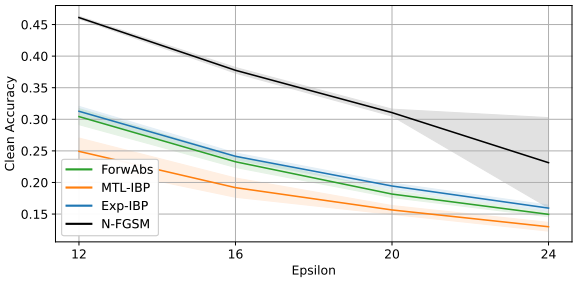


(c) `PreactResNet18`, CIFAR-10, $\epsilon = 24/255$, setup of Figure 4.8a

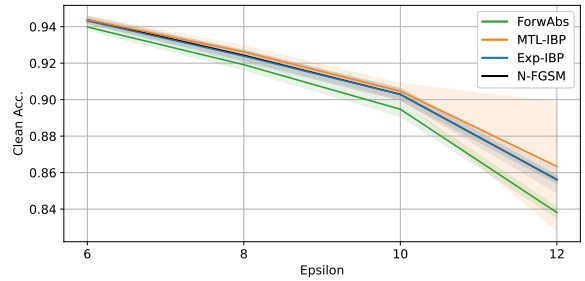
Figure 4.18: Trade-offs between estimated per-epoch runtime and AA accuracy on CIFAR-10 for $\epsilon = 24/255$.



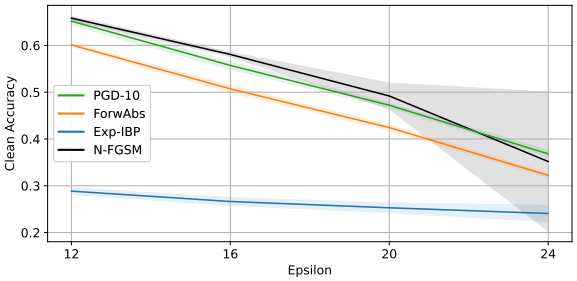
(a) `PreActResNet18`, CIFAR-10.



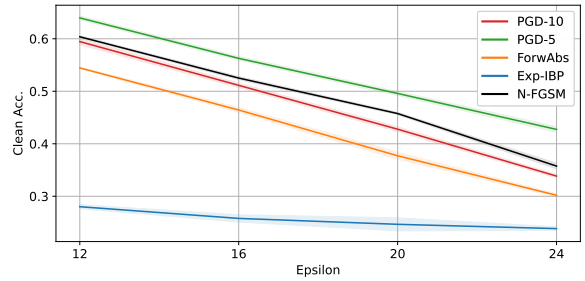
(b) `PreActResNet18`, CIFAR-100.



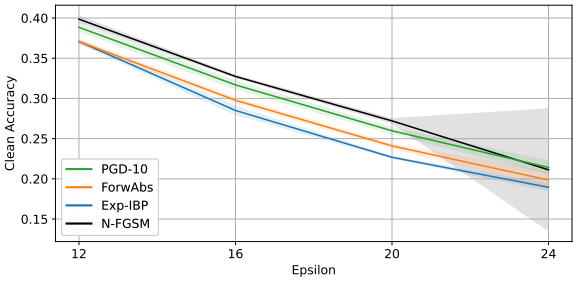
(c) `PreActResNet18`, SVHN.



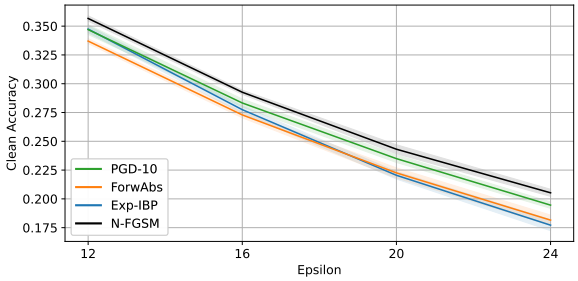
(d) `CNN-7`, CIFAR-10.



(e) `CNN-5`, CIFAR-10.

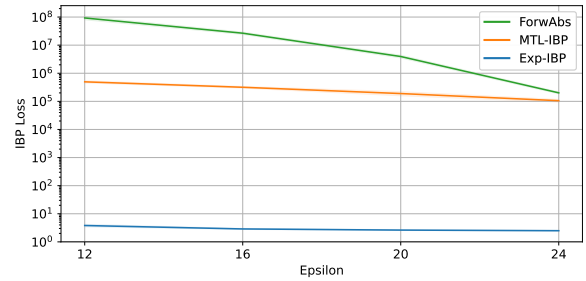


(f) `CNN-7`, CIFAR-100.

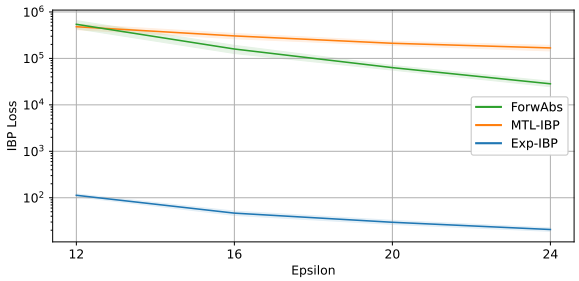


(g) `CNN-5`, CIFAR-100.

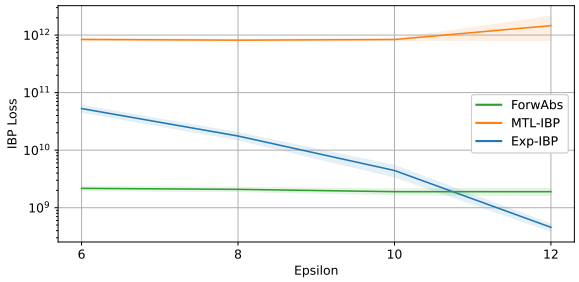
Figure 4.19: Clean accuracies for the experiments from Figure 4.8, Figure 4.9, Figure 4.12, and Figure 4.13. Means and 95% confidence intervals over 5 repetitions.



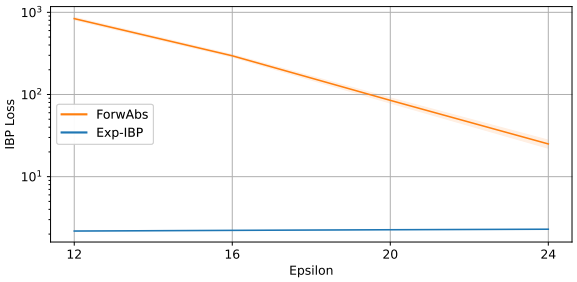
(a) `PreActResNet18`, CIFAR-10.



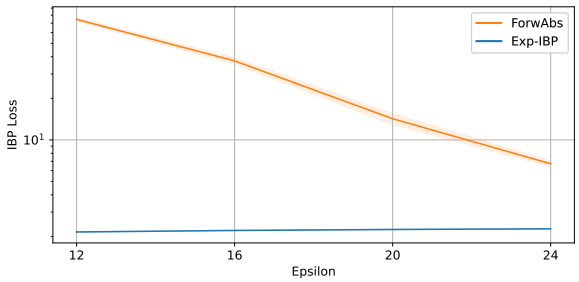
(b) `PreActResNet18`, CIFAR-100.



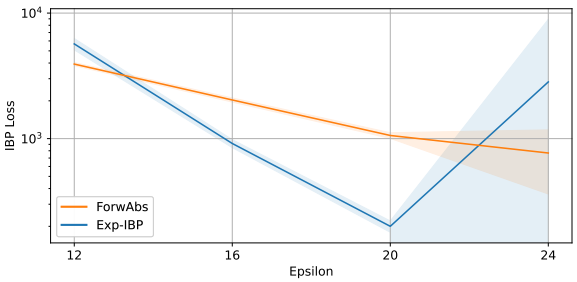
(c) `PreActResNet18`, SVHN.



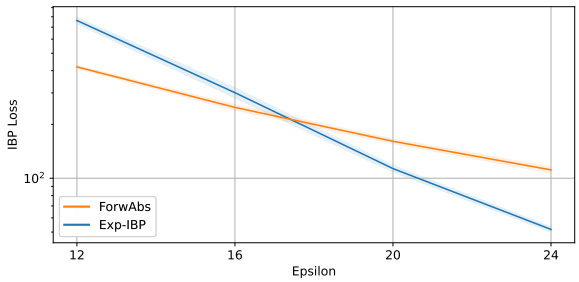
(d) `CNN-7`, CIFAR-10.



(e) `CNN-5`, CIFAR-10.



(f) `CNN-7`, CIFAR-100.



(g) `CNN-5`, CIFAR-100.

Figure 4.20: IBP losses of methods from Section 4.1 for the experiments from Figure 4.8, Figure 4.9, Figure 4.12 and Figure 4.13. Means and standard deviations over 5 repetitions. The significantly smaller IBP loss values associated with maximal AutoAttack accuracy come at a larger cost in terms of empirical robustness, resulting in worse performance trade-offs.

4.7 Related work

In addition to the work discussed in background Chapter 2 we provide a more detailed discussion of works related to this chapter.

4.7.1 SingleProp

Boopathy et al. [172] present SingleProp: a regularizer which, computed at the cost of a single network evaluation and applied on top of the clean network loss, trades certified robustness for computational efficiency. ForwAbs [5], is a conceptually-simpler alternative to SingleProp, only relying on a forward pass with the absolute weights of the network while SingleProp combines their approximated bounds with heuristic inspired from the verification literature [40, 45].
Furthermore, Boopathy et al. [172] focus on improving certified robustness exclusively, while we employ ForWabs in the context of catastrophic overfitting prevention and empirical robustness.

4.7.2 Empirical Robustness of Certified Training

Previous works have reported, explicitly or implicitly, on the empirical robustness of certified training schemes deployed to maximize certified robustness. Except for De Bartolomeis et al. [173], whose main conclusions focus on the inferior empirical robustness of the certified training schemes they considered compared to a multi-step baseline, complying with the folklore in the area, empirical robustness was not the main focus of these works.
While this was not the focus of their works, Gowal et al. [99] and Mao et al. [174] both demonstrated that certified training schemes (including IBP for Gowal et al. [99], IBP, SABR and MTL-IBP for [174]) outperform multi-step adversarial training on large perturbation radii on MNIST for relatively shallow convolutional networks. Specifically, Gowal et al. [99] showed that IBP outperforms PGD-7 training in terms of PGD-200-10 (PGD with 200 iterations and 10 restarts) accuracy for $\epsilon \in \{0.3, 0.4\}$, when using a 100-epoch training schedule. Mao et al. [174] tuned the certified training schemes to maximize certified robustness, and then positively compared their empirical robustness, measured through the attacks within MN-BAB, a complete verifier based on branch-and-bound [41], to the AA accuracy of PGD-5-3 training for $\epsilon = 0.3$ (using a 70-epoch training schedule). Moreover, De Bartolomeis et al. [173] report that, on MNIST, IBP [98] outperforms adversarial training based on a 10-step AutoPGD [79] attack in terms of AA accuracy for $\epsilon \in \{0.1, 0.2, 0.3, 0.4\}$. However, they employ inconsistent training schedules and optimizers across methods, and their adversarial training baseline severely underperforms compared to other works [174]. We omit MNIST from our main study owing to its relative simplicity, and to our focus on settings typically considered in the single-step adversarial training literature [4, 5, 7].
Furthermore, Gowal et al. [99] and Mao et al. [174] reported that certified training schemes (IBP for Gowal et al. [99], SABR and MTL-IBP for [174]) can outperform multi-step attacks in terms of empirical robustness using `CNN-7` for $\epsilon = 8/255$ on CIFAR-10. Gowal et al. [99] employ a 3200-epoch training schedule, using which PGD-7 and IBP respectively attain 34.77% and 24.95% PGD-200-10 accuracy. Mao et al. [174] rely instead on a 240-epoch schedule and again tune certified training schemes to maximize certified robustness, reporting 35.93% AA accuracy for PGD-10-3, 36.11% and 36.02% MN-BAB empirical robustness for SABR and MTL-IBP, respectively. In addition, De Bartolomeis et al. [173] reports

[172]: Boopathy et al. (2021), 'Fast Training of Provably Robust Neural Networks by SingleProp'

5: described in Section 4.1.3

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

[45]: Zhang et al. (2018), 'Efficient Neural Network Robustness Certification with General Activation Functions'

[172]: Boopathy et al. (2021), 'Fast Training of Provably Robust Neural Networks by SingleProp'

[173]: De Bartolomeis et al. (2023), 'How robust accuracy suffers from certified training with convex relaxations'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[41]: Ferrari et al. (2022), 'Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound'

[173]: De Bartolomeis et al. (2023), 'How robust accuracy suffers from certified training with convex relaxations'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

[5]: Andriushchenko and Flammarion (2020), 'Understanding and Improving Fast Adversarial Training'

[7]: Rocamora et al. (2024), 'Efficient local linearity regularization to overcome catastrophic overfitting'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

better AA accuracy for IBP [98] (32.5%), using a 160-epoch schedule, compared to 10-step AutoPGD [79] (30.2%) on CIFAR-10 for $\epsilon = 8/255$ using a residual network. However, as for their MNIST experiments, the comparison employs different training schedules and optimizers for the various methods, and features an empirically weak adversarial training baseline. We remark that N-FGSM already attains 37.76% average AA accuracy using the same network architecture in Table 4.2, outperforming the results reported above for both adversarial and certified training, and highlighting the importance of strong baselines and appropriate regularization. In fact, differently from Mao et al. [174], who adopt ℓ_1 regularization also for the adversarial training baselines, we use weight decay (with coefficient 5×10^{-5}), which is more commonly adopted in the adversarial training literature to combat robust overfitting.
Given the gap between the best empirical and certified defenses across various setups [8, 79, 125], certified training schemes are commonly understood to be at odds with empirical accuracy. Expanding on the above results and complementing Section 4.3, Section 4.4 studies whether techniques from Section 4.1.2, *when explicitly tuned for the purpose*, can improve empirical robustness compared to multi-step PGD on larger perturbation radii, different datasets, and shorter training schedules.

4.7.3 Catastrophic Overfitting in Certified Training Setups

The strong certified robustness results reported in previous work can be employed to draw conclusions on catastrophic overfitting, exploiting the fact that certified accuracy lower bounds the empirical accuracy to any attack. It is easy to conclude that any certified training scheme which lacks an adversarial training component (see Section 2.2.3) is trivially immune to CO [98, 99, 107]. Nevertheless, these are neither applicable in all training contexts (see Section 4.1.2) nor necessarily associated to the best performance [105]. For $\alpha < 1$ tuned for certified robustness via verifiers based on branch-and-bound, the results reported in [8] imply the absence of CO on the relative CC-IBP, MTL-IBP and Exp-IBP runs. As reported by De Palma et al. [8, appendix G.9], this is in spite of the fact that they rely on a randomized attack that always lands on a corner of the perturbation region and was shown to display CO by previous work in a different experimental setup [6]. Nevertheless, it is unclear from the original experiments whether $\alpha = 0$ would display CO for those experimental settings and, hence, whether expressive losses prevent it. Fully demonstrating that the reported results imply the ability of expressive losses to prevent CO requires showing that the underlying one-step attack ($\alpha = 0$) would display CO in the specific experimental setting. We here carry out such investigation, whose results are provided in Table 4.6. In particular, we compare the average AutoAttack accuracy from RS-FGSM $\eta = 10.0\epsilon$, the one-step attack employed in most expressive loss results from De Palma et al. [8][6] with the best AutoAttack accuracy obtained from the published expressive losses checkpoints from De Palma et al. [8]. Table 4.6. shows that the one-step attack displays systematic CO on MNIST with $\epsilon = 0.3$ and on CIFAR-10 with $\epsilon = 8/255$, and some signs of CO on MNIST with $\epsilon = 0.1$. In all cases, this is effectively prevented by the expressive losses runs from the literature. Nevertheless, mirroring what is outlined in Section 4.7.2, we point out that the empirical robustness of literature models is still disappointing if compared with stronger one-step baselines. For CIFAR-10 with $\epsilon = 8/255$, N-FGSM attains 37.76% average AA accuracy on the same network (Table 4.2).
In Section 4.3, we systematically investigate CO on settings popular in the relevant single-step adversarial training literature: deeper networks,

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[173]: De Bartolomeis et al. (2023), 'How robust accuracy suffers from certified training with convex relaxations'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

[125]: Jovanović et al. (2022), 'On the Paradox of Certified Training'

[98]: Shi et al. (2021), 'Fast Certified Robust Training with Short Warmup'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

[105]: Müller et al. (2023), 'Certified Training: Small Boxes are All You Need'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

6: except on CIFAR-10 with $\epsilon = 2/255$, for which De Palma et al. [159] use a multi-step attack.

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

	MNIST $\epsilon = 0.1$	MNIST $\epsilon = 0.3$	CIFAR-10 $\epsilon = 2/255$	CIFAR-10 $\epsilon = 8/255$	TinyImageNet $\epsilon = 1/255$
Method	AA acc. [%]	AA acc. [%]	AA acc. [%]	AA acc. [%]	AA acc. [%]
RS-FGSM $\eta = 10.0\epsilon$	83.84 ± 39.02	0.00	74.46 ± 0.44	0.00	28.92 ± 0.17
Best Expressive Loss	98.48	94.02	69.33	36.50	28.46

Table 4.6: CO study on `CNN-7` setups from the certified training literature. We report mean and 95% over 5 runs for the one-step attack used in most expressive loss results from De Palma et al. [8], and compare it with the best AutoAttack accuracy across the relative published CC-IBP, MTL-IBP and Exp-IBP checkpoints [8].

different underlying attacks and datasets, shorter training schedules, and larger perturbation radii.

4.7.4 Comparison with our work

Differently from the previous literature, this work, in Section 4.3, presents a systematic analysis of the ability of certified training schemes to prevent CO on setups from the single-step adversarial training literature, and in Section 4.4 extends the study by examining settings where certified training schemes relying on single-step attacks can overcome multi-step attacks.
Owing to the strong sensitivity of CO and of expressive losses (see Table 4.1) to the specific experimental setup, the conclusions in Section 4.3 do not trivially follow neither from the previous literature, nor from our study in Section 4.7.3. For instance, Figure 4.12 shows that the occurrence of CO depends on the network architecture even for the same training schedule, dataset, and perturbation radius, and a comparison with Table 4.2 illustrates the effect of changes in the training schedule. Compared to the literature discussed in Section 4.7.2, Section 4.4 provides a fair and systematic comparison (relying on the same strong attack suite [79], which is also used as tuning metric on validation) which includes larger perturbation radii (up to $\epsilon = 24/255$), the harder CIFAR-100 dataset, and a shorter cyclic training schedule more common in the adversarial training literature.
Crucially, throughout Section 4.4, rather than simply assessing the robustness of certified training schemes deployed to maximize certified robustness, we also tune these algorithms *for* empirical robustness. This can make a difference in practice. For instance, Exp-IBP attains 38.44% average AA accuracy in our experiments for $\epsilon = 8/255$ with $\alpha = 5 \times 10^{-3}$ on CIFAR-10, outperforming N-FGSM and PGD-5 from Table 4.2, the results from Table 4.6 with $\alpha = 0.5$, and all values reported by Mao et al. [174].

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

[174]: Mao et al. (2025), 'CTBench: A Library and Benchmark for Certified Training'

4.8 Conclusion

We presented a comprehensive empirical study on the utility of recent certified training techniques for empirical robustness, as opposed to verifiability, their original design goal. In particular, we showed that Exp-IBP can prevent catastrophic overfitting on single-step attacks for a variety of settings, outperforming some multi-step baselines in a subset of these. Furthermore, we presented a conceptually simple regularizer on the size of network over-approximations, named ForwAbs, that can achieve similar effects to Exp-IBP on top of single-step attacks while cutting down its overhead. While we believe that these results highlight the potential of certified training as an empirical defense, they also show the severe limitations of current techniques on harder datasets, ultimately calling for the development of better certified training algorithms.

Certified Training for Formal Explainability

5

Formal explainability is an active research area [175], with scalability as one of its main challenges. Prior works [71, 72, 176–178] have addressed this by improving explanation algorithms. Inspired by advances in certified training for verifiable robustness [8, 99, 179–181] and empirical robustness, discussed in Chapter 4, we instead aim to improve scalability by training models that are easier to explain formally.

To align with formal explanation definitions, we introduce Feature Subset Certified Training (FSCT), which trains models under partial perturbations restricted to subsets of input features. We evaluate and compare the formal explainability of FSCT trained models not only by the size of the explanations but also by the proportion of samples for which the explanations are non-trivial. We also propose a new traversal order for the VeriX algorithm [71] to compute formal explanations, extending the heuristic of Chapter 3 to the ranking of features for formal explanations. Preliminary results indicate that FSCT can improve the trade-off between explanation size and the rate of empirically ϵ-formally explainable inputs compared to other robust training methods.

5.1 Formal explainability

We have until now focused on the question of robustness (empirical or verifiable) of neural networks. We now turn to another important aspect of trustworthy AI: explainability. The goal of the field of explainable AI (XAI) is to provide human-interpretable explanations of the decisions of a model. The field is vast, we refer to Molnar [182] for a general introduction to the field or Mersha et al. [183] for a survey on the topic.

We focus in this chapter on the problem of finding the most relevant features for the decision of a trained neural network f_θ on a given input $\boldsymbol{x}$. The family of XAI methods in this setting is the family of **post-hoc** methods, computing **local explanations**: they are applied on trained models and on a single input sample.

We first present briefly some of the existing heuristic methods for feature attribution and then focus on the recent line of work on **formally robust explanations**. After introducing the relevant definitions and the VeriX algorithm [71] to compute them, we discuss their limits and the need for assessing the (empirical) formal explainability of models.

5.1.1 Feature attribution methods

Throughout this chapter we consider $f_\theta : \mathbb{R}^{h\times w\times c} \to \mathbb{R}^K$ a neural network trained on samples $(\boldsymbol{x}, y) \sim \mathcal{D}$ with $\mathcal{D}$ a multi-class image classification dataset. Given an input sample $\boldsymbol{x} \in \mathbb{R}^{h\times w\times c}$ the model outputs a vector of logits $f_\theta(\boldsymbol{x}) \in \mathbb{R}^K$. The predicted label is given by $y_{\text{pred}} = \text{argmax}_k\, f_\theta(\boldsymbol{x})[k]$.

Remark 5.1.1 (Input features) For explanations of images the input features are often considered spatially (*e.g.* points or subsets in a $h \times w$ grid) but aggregated across channels. In this case the feature space (from an explanation point of view) is of dimension $d = w \times h$ for an image of width w and height h, independently of the number of

[175]: Marques-Silva (2022), 'Logic-Based Explainability in Machine Learning'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[72]: Bassan and Katz (2023), 'Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks'

[176]: Wu et al. (2024), 'Better Verified Explanations with Applications to Incorrectness and Out-of-Distribution Detection'

[177]: Doncenco et al. (2025), 'A Dive into Formal Explainable Attributions for Image Classification'

[178]: Izza et al. (2024), 'Distance-restricted explanations: theoretical underpinnings & efficient implementation'

channels, even if in practice the model operates on a space of dimension $w \times h \times c$ with c the number of channels.

Feature attribution methods assign a score to each feature of the input sample $\boldsymbol{x}$, representing its importance for the decision of the model f_θ on $\boldsymbol{x}$. We briefly present some of the most popular feature attribution methods.
SHAP [184] studies the problem under the scope of game theory and proposes to approximate the Shapley values of the features. LIME [69] proposes to fit a simple interpretable model (typically a linear model) locally around the input sample $\boldsymbol{x}$ and use the coefficients of this model as feature importance scores. Integrated Gradients [66] compute the average gradients of the model output with respect to the inputs along a linear path from a baseline input (typically the zero vector) to the input sample $\boldsymbol{x}$. The gradients are then used as feature importance scores. SmoothGrad [67] compute average gradients with respects to inputs over samples obtained by adding small Gaussian noise to the input $\boldsymbol{x}$. Grad-CAM [185] propose to use gradient with respects to the last convolution layer, averaging them over the filters and upsampling to the input dimension. Backpropagation methods, such as Layer-wise Relevance Propagation [186], Guided Backpropagation [187] or DeepLift [188], start from the output and use different rules to compute attributions between consecutive layers through the network until reaching the input. They have been shown to be equivalent to gradient-based methods Ancona et al. [189] under some conditions.
All those methods are heuristic in nature and provide no guarantee on the relations between the scores and the actual importance of the feature for the decision of the model [190]. Evaluating and comparing them is an open problem [191–193].

5.1.2 Formally robust explanations

To address this lack of guarantees, a line of work proposes to use formal methods to classify features as relevant when they must remain fixed to preserve the model's decision, or irrelevant when any change in their value does not alter that decision. Early works include Shih et al. [194], who introduced the notion of **prime implicant explanations** for Bayesian network classifiers, and Ignatiev et al. [195], who proposed computing **abductive explanations** for neural networks with up to 20 neurons in a single hidden layer with ReLU activations. A broader survey covering models beyond neural networks can be found in Marques-Silva [175]. These works draw inspiration from propositional logic, where variables not part of the explanations can take any boolean value. Applied to neural networks, the irrelevant features can take any value in the network's input space, *e.g.* $[0, 1]$ for scaled image inputs. This hinders the applicability of such methods, as proving robustness when the perturbation space is the entire input space is quite challenging.
To address this limitation, Huang and Marques-Silva [70] and Wu et al. [71] propose **distance-based formal explanations**. In this setting, features are deemed irrelevant if perturbing them within a ball of radius ϵ does not change the model's prediction. By connecting the problem of finding ϵ-formal explanations to neural network verification, these methods can leverage off-the-shelf verifiers. La Malfa et al. [196] adopt a similar approach for NLP models, where the perturbation space is defined as a bounded box enclosing word embeddings rather than a ball.
We present here in particular the formalism and the algorithm of Wu et al. [71], which form the basis of our approach.

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[99]: Gowal et al. (2018), 'On the effectiveness of interval bound propagation for training verifiably robust models'

[179]: Shi et al. (2021), 'Fast certified robust training with short warmup'

[180]: Mirman et al. (2019), 'A Provable Defense for Deep Residual Networks'

[181]: Müller et al. (2022), 'The Third International Verification of Neural Networks Competition (VNN-COMP 2022): Summary and Results'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[182]: Molnar (2025), *Interpretable Machine Learning*

[183]: Mersha et al. (2024), 'Explainable artificial intelligence: A survey of needs, techniques, applications, and future direction'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[184]: Lundberg and Lee (2017), 'A unified approach to interpreting model predictions'

[69]: Ribeiro et al. (2016), '" Why should i trust you?" Explaining the predictions of any classifier'

[66]: Sundararajan et al. (2017), 'Axiomatic attribution for deep networks'

[67]: Smilkov et al. (2017), *SmoothGrad: removing noise by adding noise*

[185]: Selvaraju et al. (2017), 'Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization'

[186]: Bach et al. (2015), 'On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation'

[187]: Springenberg et al. (2015), *Striving for Simplicity: The All Convolutional Net*

[188]: Shrikumar et al. (2017), 'Learning important features through propagating activation differences'

[189]: Ancona et al. (2018), 'Towards better understanding of gradient-based attribution methods for Deep Neural Networks'

[190]: Narodytska et al. (2019), 'Assessing Heuristic Machine Learning Explanations with Model Counting'

Definition 5.1.1 (ϵ-formally robust explanations) *Given a sample* $(\boldsymbol{x}, y) \sim \mathcal{D}$, *a neural network* f_θ *and a perturbation budget* $\epsilon > 0$, ***irrelevant features*** *are a subset of features of* $\boldsymbol{x}$, *indexed by a subset* $\mathcal{I} \subseteq [\![d]\!]$ *such that any perturbation of* $\boldsymbol{x}[\mathcal{I}]$[1] *does not alter the classification of* f_θ *on* $\boldsymbol{x}$. *More formally:*

$$\forall \boldsymbol{x}', \|\boldsymbol{x}'[\mathcal{I}] - \boldsymbol{x}[\mathcal{I}]\| < \epsilon \wedge \boldsymbol{x}'[\mathcal{I}^c] = \boldsymbol{x}[\mathcal{I}^c] \\ \implies \operatorname*{argmax}_k f_\theta(\boldsymbol{x}')[k] = \operatorname*{argmax}_k f_\theta(\boldsymbol{x})[k]. \tag{5.1}$$

$\mathcal{I}$ is said to be ***subset-optimal*** *if allowing to perturb any additional feature makes it possible to find a perturbation successfully changing the decision of the network:*

$$\forall i' \in \mathcal{I}^c, \exists \boldsymbol{x}' : \|\boldsymbol{x}'[\mathcal{I} \cup \{i'\}] - \boldsymbol{x}[\mathcal{I} \cup \{i'\}]\| \leq \epsilon \\ \bigwedge \boldsymbol{x}'[\{\mathcal{I} \cup \{i'\}\}^c] = \boldsymbol{x}[\{\mathcal{I} \cup \{i'\}\}^c] \\ \bigwedge \operatorname*{argmax}_k f_\theta(\boldsymbol{x}')[k] \neq \operatorname*{argmax}_k f_\theta(\boldsymbol{x})[k]. \tag{5.2}$$

Given a set $\mathcal{I}$ of irrelevant features its complement $\mathcal{R} = [\![d]\!] \setminus \mathcal{I}$ is a ***ϵ-formally robust explanation***. *Furthermore, we say that $\mathcal{R}$ is a minimal explanation if $\mathcal{I}$ is maximal.*

1: If $\boldsymbol{x}$ is an image with width w and height h, the feature space is of dimension $d = w \times h$. The perturbations are applied across all channels.

Intuitively, $\mathcal{R}$ represents the *sufficient* set of features explaining the decision of f_θ, since perturbing all other features does not affect it. Multiple subset-minimal robust explanations may exist for a given sample $\boldsymbol{x}$, and in the worst case, their number can grow exponentially with the input dimension [197].

Remark 5.1.2 (Empty explanations) If f_θ is robust over the entire ϵ-ball centered at $\boldsymbol{x}$ (*i.e.*, $\mathcal{I} = [\![d]\!]$), then all features are irrelevant, and the corresponding explanation is empty: $\mathcal{R} = \emptyset$.

5.1.3 Computing formal explanations

The definition of irrelevant features in Eq. (5.1) corresponds to a local robustness property, similar to Definition 2.2.2, except that the allowed perturbations are restricted to a subset of features. Irrelevant features can be computed using an off-the-shelf verifier.

The VeriX algorithm, introduced by Wu et al. [71], computes subset-minimal explanations using a greedy approach and the verifier Marabou [147]. The procedure is summarized in Algorithm 1.

The algorithm employs three sub-routines: TraversalOrder, Perturb, and Verify. TraversalOrder relies on heuristics to estimate the influence of each input feature on the output $f_\theta(\boldsymbol{x})$ and returns the features ordered by their estimated importance. Formally σ is a permutation of $(0, \dots d-1)$. While VeriX is guaranteed to return a subset-maximal set of irrelevant features, its size depends on the ordering σ. An effective heuristic for TraversalOrder should yield irrelevant sets of large cardinality. We discuss in Section 5.3 different algorithms from the literature, and our own proposed ordering, inspired by the ReCIPH heuristic from Chapter 3.

Perturb instantiates the set ϕ of allowed perturbations at every step:

$$\textsc{Perturb}(\boldsymbol{x}, \mathcal{S}, \epsilon) := \{\boldsymbol{x}' : \|\boldsymbol{x}'[\mathcal{S}] - \boldsymbol{x}[\mathcal{S}]\| < \epsilon \wedge \boldsymbol{x}'[\mathcal{S}^c] = \boldsymbol{x}[\mathcal{S}^c]\} \tag{5.3}$$

[191]: Adebayo et al. (2018), 'Sanity checks for saliency maps'

[192]: Xu-Darme et al. (2023), 'On the stability, correctness and plausibility of visual explanation methods based on feature importance'

[193]: Hedström et al. (2023), 'The Meta-Evaluation Problem in Explainable AI: Identifying Reliable Estimators with MetaQuantus'

[194]: Shih et al. (2018), 'A symbolic approach to explaining bayesian network classifiers'

[195]: Ignatiev et al. (2019), 'Abduction-based explanations for machine learning models'

[175]: Marques-Silva (2022), 'Logic-Based Explainability in Machine Learning'

[70]: Huang and Marques-Silva (2023), 'From Robustness to Explainability and Back Again'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[196]: La Malfa et al. (2021), 'On Guaranteed Optimal Robust Explanations for NLP Models'

More specifically, Bassan et al. [197] show that for a boolean feature space: $\boldsymbol{x} \in \{0,1\}^d$ there exists f_θ and $\boldsymbol{x}$ such that there are $\Theta(\frac{2^d}{\sqrt{d}})$ subset minimal robust explanations. The statement also holds for the number of cardinally-minimal robust explanations.

[147]: Katz et al. (2019), 'The Marabou Framework for Verification and Analysis of Deep Neural Networks'

```
Input: x, ϵ, f_θ
Output: I, R

I ← ∅, R ← ∅
σ ← TraversalOrder(f_θ, x)
y_pred ← argmax_k f_θ(x)[k]
for i in σ do
    ϕ ← Perturb(x, I ∪ {i}, ϵ)
    res ← Verify(ϕ, f_θ, y_pred)
    if res then
        I ← I ∪ {i}
    else
        R ← R ∪ {i}
    end if
end for

return I, R
```

Algorithm 1: VeriX algorithm to compute subset-minimal robust explanations.

with $\mathcal{S} = \mathcal{I} \cup \{i\}$ the union of the set of previously proven irrelevant features $\mathcal{I}$ with the current candidate feature $\{i\}$.
Finally, Verify takes a set of perturbations, a model f_θ and its predicted label[2], and returns *True* if no perturbation changes the classification of f_θ and *False* otherwise.
Now that we have an algorithm to compute formal explanations we can define a notion of **formally explainable models**.

2: Notice that the verification query is slightly different from classically verifying local robustness: we do not consider the true label y but rather the predicted label $y_{\text{pred}} = \text{argmax}_k f_\theta(\boldsymbol{x})[k]$. Even when the model misclassifies the inpu, we want to explain its decision.

Definition 5.1.2 (ϵ-formally explainable) *Let f_θ be a neural network with parameters θ, and $\boldsymbol{x} \sim \mathcal{D} \in \mathbb{R}^d$ a sample with predicted label $y_{pred} = \text{argmax}_k f_\theta(\boldsymbol{x})[k]$. Let $\epsilon > 0$ be a perturbation budget and σ an ordering of the input features.*
*We say that f_θ is **ϵ-formally explainable** for the input $\boldsymbol{x}$ if VeriX returns a set of irrelevant features $\mathcal{I}$ such that $0 < |\mathcal{I}| < d$.*

Given some perturbation budget ϵ, the explainability of a model can be evaluated both by the average size of the explanations and by the number of samples for which the model is formally explainable. Changing ϵ is tempting as lowering it makes the verification easier and should lead to larger sets of irrelevant features. However, we will see in Section 5.4.3 that it can lead to poor explainability in practice due to the larger number of non ϵ-formally explainable samples.

5.1.4 Limits of optimal robust explanations

The problem of verifying the local robustness of neural networks has been shown to be NP-Hard for feedforward models with ReLU activation functions [134]. The implementation of Verify in VeriX uses the verifier Marabou [198], which is capable of checking whether such properties hold. For a d-dimensional feature space, d calls to Verify are necessary. This greatly limits the scalability of VeriX. The network evaluated in Wu et al. [176] has 10 hidden neurons and a test accuracy of 92.26% on the MNIST dataset where models now easily achieve accuracy greater than 99%.[3] The perturbation budget used ($\epsilon = 0.05$) is also much lower than those typically used in robustness work ($\epsilon \in \{0.1, 0.3\}$).
Our goal in this work is to scale formal explanations to large neural networks with perturbation budgets ϵ comparable to those employed in the robustness literature.

3: The models adversarially trained and even some of the models trained with certified training in Mao et al. [174] have more than 99% accuracy, up to 99.58%.

VeriX does not scale in such a setting. In fact, the underlying solver Marabou [176] fails to parse the `CNN-7` network used commonly in the certified training literature.

5.1.5 Using incomplete but sound verifiers

```
Input: x, ϵ, f_θ
Output: I, R, U

I ← ∅, R ← ∅, U ← ∅
σ ← TraversalOrder(f_θ, x)
y_pred ← argmax_k f_θ(x)[k]
for i in σ do
  ϕ ← Perturb(x, I ∪ {i}, ϵ)
  res ← Verify(ϕ, f_θ, y_pred)
  if res is True then
      I ← I ∪ {i}
  elseif res is False then
      R ← R ∪ {i}
  else # res is unknown
      U ← U ∪ {i}
  end if
end for

return I, R, U
```

Algorithm 2: VeriX-incomplete

A straightforward way to improve the scalability of VeriX is to use a faster, *sound* but *incomplete* verifier for the Verify procedure. Many approaches have been proposed to improve the scalability of neural network verification such as the linearization techniques discussed in Section 2.4. When the Verify call neither proves nor falsifies the property, it can now return *unknown* and a corresponding set $\mathcal{U}$ is maintained and returned at the end of the algorithm. The modifications of VeriX are shown in blue in Algorithm 2. This version of the algorithm remains *sound*: no perturbation of features in $\mathcal{I}$ will change the decision of f_θ. $\mathcal{I}$ is now not necessarily minimal, but it provides a lower bound on the maximal set of irrelevant features, given an ordering σ:

Proposition 5.1.1 *Let f_θ be a neural network, $\boldsymbol{x} \sim \mathcal{D}$ a sample, $\epsilon > 0$ a perturbation budget. If VeriX-incomplete returns a set $\mathcal{I}$ of irrelevant features and VeriX returns a set $\mathcal{I}'$ of irrelevant features, **using the same ordering** σ, we have:*

$$|\mathcal{I}| \leq |\mathcal{I}'| \tag{5.4}$$

In fact we have:

$$\mathcal{I} \subseteq \mathcal{I}' \tag{5.5}$$

Directly using such an approach provides limited information on standard networks. Linearization techniques are not precise enough to assess the robustness when using perturbation budgets ϵ typically used in the robustness literature.
Another limit of this approach is the difficulty to find counterfactuals. At the end of the execution of VeriX-incomplete, the set $\mathcal{U}$ contains features that could not be proven irrelevant but the set of relevant features $\mathcal{R}$ remains empty. Linearization techniques are not precise enough to prove the entire perturbed space leads to a different decision. Attempting to find counterexamples using adversarial attacks also fails: the search space is constrained to a subspace of the entire feature space ($\mathcal{I} \cup \{i\}$) where

all but one feature were already proven to be irrelevant.
We introduce in Section 5.2, a training procedure to enable the use of incomplete verifiers based on over-approximations and bound propagation. While it improves the number of verifiably irrelevant features, proving features to be relevant remains intractable. We now describe the notion of empirically formally explainable models to evaluate the explainability of a model in a practical setting.

5.1.6 Empirically formally explainable models

The motivation to induce some robustness at training time to produce formally explainable models is clear from the definition of formal explanations(see Definition 5.1.1). However, the risk is to obtain models that cannot be assessed to be ϵ-formally explainable (Definition 5.1.2) as we are restricted to incomplete verifiers for scalability reasons. We introduce the following notion of **empirically ϵ-formally explainable models** to precise the explainability of a model f_θ on a dataset $\mathcal{D}$ with respect to a perturbation budget ϵ:

The definition relies on the choice of the adversarial attack generator. In practice, we use AutoAttack [79], a family of strong attacks used commonly in the robustness literature. We emphasize that the attack aims to find a perturbation successfully changing the decision of the network, even when the network prediction is not the true label.

Definition 5.1.3 (Empirically ϵ-formally explainable) *Let $f_\theta \in \mathcal{F}$ be a neural network, with $\mathcal{F}$ the family of classifier, and $\mathcal{D} \subset (\mathcal{X} \times \mathcal{Y})$ a dataset for multi-class classification.*

Let $\mathbb{A} : \mathcal{X} \times \mathcal{Y} \times \mathcal{F} \times \mathbb{R}^+ \to \mathcal{X}$ be an adversarial attack generator: $\mathbb{A}(\boldsymbol{x}, y_{pred}, f_\theta, \epsilon) = \boldsymbol{x}_{adv} \in B(\boldsymbol{x}, \epsilon)$ is an adversarial perturbation of $\boldsymbol{x}$ computed to change prediction $y_{pred} = \text{argmax}_k\, f_\theta(\boldsymbol{x})[k]$.

*We say that f_θ is **empirically ϵ-formally explainable** for the input $\boldsymbol{x}$ and the oracle $\mathbb{A}$ if the algorithm VeriX-incomplete returns a set of irrelevant features $\mathcal{I}$ such that: $0 < |\mathcal{I}| < d$ and $\text{argmax}_k\, f_\theta(\boldsymbol{x}_{adv})[k] \neq y_{pred}$.*

Informally, f_θ being empirically ϵ-formally explainable on $\boldsymbol{x}$ for an oracle $\mathbb{A}$ means that we know the set of irrelevant features is not trivial. In particular, **it cannot be full** since we have found a perturbation successfully changing the decision of f_θ, implying the existence of relevant features.

5.2 Training for Formal Explainability

We consider three training procedures to obtain empirically formally explainable models: adversarial training, certified training using expressive losses [8] and a novel training procedure we call **Feature Subset Certified Training** (FSCT).

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

5.2.1 Feature Subset Certified Training (FSCT)

Motivated by the definition of irrelevant features in Definition 5.1.1, we propose a training procedure that considers perturbations *on a subset of input features only*.

Remark 5.2.1 (Relation to Expressive Losses) This notion of partial robustness is different from the approach of SABR described in Section 2.3.1. SABR computes a subset of the perturbation space by *considering a ball with a smaller radius*. Here we consider a subspace of the perturbation space where some features are kept constant.

In general the expressive losses can be seen as a different way of loosening the certified robustness goal. We will show in Section 5.4 how they compare to FSCT for formal explainability.

Let $\mathcal{D} = \{(\boldsymbol{x}^{(0)}, y^{(0)}) \dots (\boldsymbol{x}^{(n-1)}, y^{(n-1)})\}$, be a classification dataset with $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$, and $y^{(i)} \in \{0 \dots K-1\}$ for all $(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D}$.
We consider a sample $\boldsymbol{x} \in \mathbb{R}^d$, a perturbation budget $\epsilon > 0$, a binary mask $\boldsymbol{m} \in \{0,1\}^d$ that represents a subset of perturbed features.

We consider here the case where $d = h \times w \times c$ with $c = 1$ for simplicity. When $c > 1$ the perturbations are applied across all channels.

Definition 5.2.1 (Masked perturbation set) *$M(\boldsymbol{x}, \boldsymbol{m}, \epsilon) \subset \mathbb{R}^d$ is a subset of $\mathbb{R}^d$ parametrized by $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{m} \in \{0,1\}^d$ and $\epsilon > 0$ defined as follows:*

$$M(\boldsymbol{x}, \boldsymbol{m}, \epsilon) := \left\{ \boldsymbol{x}' \in \mathbb{R}^d : \begin{array}{ll} \boldsymbol{x}'[i] = \boldsymbol{x}[i] & \text{if } \boldsymbol{m}[i] = 1, \\ |\boldsymbol{x}'[i] - \boldsymbol{x}[i]| \leq \epsilon & \text{if } \boldsymbol{m}[i] = 0 \end{array} \right\}. \tag{5.6}$$

We define subset certified robustness similarly to Definition 2.2.3:

Definition 5.2.2 (Feature Subset Certified Robustness) *A network f_θ is said to be **feature subset certifiably robust** on an input sample $(\boldsymbol{x}, y) \sim \mathcal{D}$ if and only if the difference between the ground-truth loit $f_\theta(\boldsymbol{x}')[y]$ and the other logits is positive for all $\boldsymbol{x}' \in M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$. We define the worst-case logit difference on the masked pertubation set $M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$ as:*

$$\boldsymbol{z}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y} := \min_{\boldsymbol{x}' \in M(\boldsymbol{x},\boldsymbol{m},\epsilon)} \left[\boldsymbol{z}_{f_\theta}(\boldsymbol{x}', y) \right]. \tag{5.7}$$

We recall that $\boldsymbol{z}_{f_\theta}(\boldsymbol{x}, y) := f_\theta(\boldsymbol{x})[y] - f_\theta(\boldsymbol{x})$ is the vector of logit differences.

f_θ is feature subset certifiably robust on $(\boldsymbol{x}, y)$ if and only if:

$$\forall i \neq y,\ \boldsymbol{z}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}[i] > 0. \tag{5.8}$$

We can formulate a verified loss, given optimal bounds of the logit differences on the masked perturbation set $M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$, $\boldsymbol{z}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}$:

$$\mathcal{L}_{FSCT}(\theta, \boldsymbol{x}, y, \boldsymbol{m}, \epsilon) = \mathcal{L}(-\boldsymbol{z}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}, y) \tag{5.9}$$

where $\mathcal{L}$ is a standard classification loss such as the cross-entropy loss.
We can use bound propagation techniques to compute lower bounds $\underline{\boldsymbol{z}}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}$, analogously to Section 2.2.3 of the worst-case logit differences on the masked perturbation set $M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$: $\underline{\boldsymbol{z}}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y} \leq \boldsymbol{z}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}$ and use them to define an approximated verified loss for feature subset certified training:

$$\mathcal{L}_{\underline{\text{ver}}}(\theta, \boldsymbol{x}, y, \boldsymbol{m}, \epsilon) = \mathcal{L}(-\underline{\boldsymbol{z}}_{f_\theta}^{M(\boldsymbol{x},\boldsymbol{m},\epsilon),y}, y). \tag{5.10}$$

This loss can be used in conjunction with an adversarial loss similarly as the expressive losses discussed in Example 2.3.1. The adversary can be computed using gradient-based attacks constrained to the masked perturbation set $M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$.

Identifying the subsets to use during training A key component of FSCT is the choice of the binary masks (*i.e.* the subsets) used to define the masked pertubation set $M(\boldsymbol{x}, \boldsymbol{m}, \epsilon)$. We seek masks that:

- Select *important* features to be kept constant.

- Do not destabilize the training dynamics.

[65]: Simonyan et al. (2014), *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

[66]: Sundararajan et al. (2017), 'Axiomatic attribution for deep networks'

[67]: Smilkov et al. (2017), *SmoothGrad: removing noise by adding noise*

Motivated by the works in the feature attribution literature [65–67] we propose to rank the features using the input gradients of a model computed at some epoch τ, and select the top k features to be kept constant. For a vector $\boldsymbol{v} = (v_1 \dots v_d) \in \mathbb{R}^d$ and an integer $k \leq d$ we denote by $\text{ArgBot}_k(\boldsymbol{v})$ the set of k indices of the lowest components of $\boldsymbol{v}$:

$$\text{ArgBot}_k(\boldsymbol{v}) := \{i_1 \dots i_k\} \subseteq [\![d]\!] \text{ such that:} \tag{5.11}$$
$$v_{i_1} \leq v_{i_2} \leq \dots \leq v_{i_k} \text{ and} \tag{5.12}$$
$$v_j \geq v_{i_k} \text{ for any } j \notin \{i_1 \dots i_k\}. \tag{5.13}$$

such that: $v_{i_k} \leq v_{i_{k-1}} \leq \dots \leq v_{i_1}$ and $\{i_1 \dots i_k\} \subseteq [\![d]\!]$.
To compute our mask we first average the absolute values of the input gradients channel-wise, matching the definition of irrelevant features. If $d = h \times w \times c$ we compute

$$\boldsymbol{r} = \frac{1}{c} \sum_{c=1}^{c} \left| \frac{\partial f_\theta(\boldsymbol{x})[y]}{\partial \boldsymbol{x}_{:,:,c}} \right| \tag{5.14}$$

with $\boldsymbol{x}_{:,:,c} \in \mathbb{R}^{h \times w}$ the c-th channel of the input $\boldsymbol{x}$. We then sort $\boldsymbol{r}$ and chose $k = \lfloor \mu \times d \rfloor$. with $\mu \in [0, 1]$ a hyperparameter representing the percentage of features to be perturbed. The mask $\boldsymbol{m}_x$ is then defined as:

$$\boldsymbol{m}_x[i] = \begin{cases} 1 & \text{if } i \notin \text{ArgBot}_k(\boldsymbol{r}) \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

If $\mu = 0$ no feature is perturbed: the model is trained normally. If $\mu = 1$ all features are perturbed: the model is trained using standard certified training.

To promote stable training we compute the gradients and the corresponding masks at a given epoch τ and use them unchanged for the rest of the training. τ should be chosen after a few epochs to let the model learn some meaningful features but not too late to let the model adapt to the partial robustness objective.
Two hyperparameters are thus introduced: τ the epoch when we compute the masks, and μ the number of features to perturb.

We express μ as a percentage of the input features.

Before showing the experimental results of FSCT in Section 5.4 we first discuss the ordering procedure used in VeriX and VeriX-incomplete.

5.3 Traversal orders

The procedure TraversalOrder is a key component of VeriX and remains crucial in VeriX-incomplete.
We first describe three traversal orders from the literature:

- Occlusion[4]
- IBP bounds
- Gradient.

4: Also called sensitiviy in Wu et al. [71]

5.3.1 Existing orders

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

Occlusion [71] The occlusion scores are defined as

$$\sigma_{occl}(\boldsymbol{x}[i]) = f_\theta(\boldsymbol{x})[y_{\text{pred}}] - f_\theta(\boldsymbol{x}')[y_{\text{pred}}] \tag{5.16}$$

where $\boldsymbol{x}'$ is the input $\boldsymbol{x}$ with the feature i occluded (set to 0 or to 1-$\boldsymbol{x}[i]$ depending on the dataset ranges).[5] The idea is to measure the impact of occluding a feature on the logit of the predicted class.

5: Wu et al. [71] uses 1-$\boldsymbol{x}[i]$ for occlusion on MNIST and 0 for other datasets.

IBP bounds [176] The idea is to compute an approximation of the lower bound of the predicted logit *when only perturbing a single pixel*. If this lower bound is high the feature is likely to be irrelevant as the predicted logit remains high. The IBP bounds scores are formally defined as:

[176]: Wu et al. (2024), 'Better Verified Explanations with Applications to Incorrectness and Out-of-Distribution Detection'

$$\sigma_{IBP}(\boldsymbol{x}[i]) = l_{y_{\text{pred}}} \tag{5.17}$$

Where $l_{y_{\text{pred}}} \leq \min_{\boldsymbol{x}' \in M(\boldsymbol{x},\boldsymbol{m},\epsilon)} f_\theta(\boldsymbol{x}')[y_{\text{pred}}]$ is a lower bound of the predicted logit computed using IBP on a masked pertubation set with a mask $\boldsymbol{m}$ with $\boldsymbol{m}[i] = 0$ and $\boldsymbol{m}[j] = 1$ for $j \neq i$.

The scores are then sorted in descending order, unlike the other methods where a low score means the feature is likely to be irrelevant.

Gradient [177] The gradient scores are defined as:

[177]: Doncenco et al. (2025), 'A Dive into Formal Explainable Attributions for Image Classification'

$$\sigma_{grad}(\boldsymbol{x}[i]) = \left| \frac{\partial f_\theta(\boldsymbol{x})[y_{\text{pred}}]}{\partial \boldsymbol{x}[i]} \right| \tag{5.18}$$

Intuitively gradient measures the sensitivity of the output logit with respect to infinitesimal perturbations of the input feature. In general feature attribution methods are natural candidates for the ordering procedure as they provide a score of importance for each input feature. Doncenco et al. [177] studied the use of various feature attribution methods for the ordering procedure in VeriX and found no significant advantage of using more complex methods over input gradients. They also prove that for linear models the gradient ordering is optimal.

The gradient based ordering is also called Saliency.

5.3.2 Linear Coefficients as traversal order

We also introduce a new traversal order, to the best of our knowledge, using relational coefficients. Inspired by the problem of ranking input features to better guide input partitioning we described in Chapter 3, we propose to use the linear coefficients as a traversal order for VeriX.
We consider as the initial perturbation space the entire ball $B(\boldsymbol{x}, \epsilon)$. For a linearization technique $\mathbb{L}$, we write $\underline{\Lambda}_{\mathbb{L}} \in \mathbb{R}^n, \underline{b} \in \mathbb{R}$ and $\overline{\Lambda}_{\mathbb{L}} \in \mathbb{R}^n, \overline{b} \in \mathbb{R}$ the coefficient vectors and bias obtained by the analysis of the network f_θ on $B(\boldsymbol{x}, \epsilon)$ using the linearization technique $\mathbb{L}$. Using sound linearization techniques we have that:

$$\forall \boldsymbol{x} \in B(\boldsymbol{x}, \epsilon), \underline{\Lambda}_{\mathbb{L}} \cdot \boldsymbol{x} + \underline{b} \leq f_\theta(\boldsymbol{x})[y_{\text{pred}}] \leq \overline{\Lambda}_{\mathbb{L}} \cdot \boldsymbol{x} + \overline{b}. \tag{5.19}$$

We then compute the scores as the mean of the absolute values of the lower and upper bound coefficients:

$$\sigma_{coeff}(\boldsymbol{x}[i]) = \frac{|\underline{\Lambda}_{\mathbb{L}}[i]| + |\overline{\Lambda}_{\mathbb{L}}[i]|}{2}. \tag{5.20}$$

The features with the lowest scores are perturbed first as they are estimated to have the lowest influence on the output logit.
When the input space has more than one channel the scores are averaged across channels for all traversal orders.

5.3.3 Complexity of the different traversal orders

The occlusion order requires d forward passes, as each pixel is perturbed individually and similarly IBP bounds ordering requires $2 \times d$ forward passes as bound computation typically requires two forward passes (see Section 2.2.4). Both methods can be batched, using a batch size of d for inputs of dimension d, or minibatches can be used if the d is too large.
The gradient order requires a single forward and backward pass to

compute the gradients with respect to the input. The linear coefficient order complexity depends on the underlying linearization technique $\mathbb{L}$ used. When using CROWN the number of forward and backward passes is quadratic in the number of layers of the network.
In practice, we find that the gradient method is the fastest and the IBP bounds method the slowest even when batching as reported in Table 5.1. We believe that IBP bounds computation does not benefit from batching as much as occlusion due to some overhead in the Auto_LiRPA implementation. In general the time to compute the ordering is negligible compared to the verification queries in VeriX.

Table 5.1: Average time (in ms) and standard deviation to compute the different traversal orders on a `CNN-7` network on CIFAR-10. We use $\epsilon = 4/255$. The times are measured on a NVIDIA H100 GPU across the first 1000 samples of the CIFAR-10 test set. For occlusion and IBP bounds the computation is entirely batched (*i. e.* with batch size of 1024). We train the network using three different training procedures: standard training (Clean), adversarial training (PGD) and certified training with the CC-IBP loss [8].

Model	Traverse	Time (ms)
Clean	occlusion	45 ± 6
	gradient	26 ± 1
	ibp bounds	370 ± 24
	crown coefs	155 ± 5
PGD	occlusion	45 ± 6
	gradient	26 ± 6
	ibp bounds	368 ± 27
	crown coefs	158 ± 14
CC-IBP	heuristic	45 ± 6
	gradient	26 ± 1
	ibp bounds	369 ± 24
	crown coefs	184 ± 23

[176]: Wu et al. (2024), 'Better Verified Explanations with Applications to Incorrectness and Out-of-Distribution Detection'

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

With Auto_LiRPA we can use a variety of linearization-based verifiers such as CROWN and α−CROWN and run them on GPU.

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

6: The network and parameters are all taken from Wu et al. [71]

7: We use α−CROWN with 4 iterations and a step size of 0.1.

5.4 Experimental results

We first evaluate the impact of using a dichotomy search to find a large irrelevant set before running VeriX-incomplete, an improvement proposed in Wu et al. [176]. Following our observation that the initial set thus computed is very close to the final irrelevant set, we rely solely on the dichotomy search to further compare the different traversal orders. Finally, using the best traversal order for robust training and the dichotomy search, we compare three different approach to train for formal explainability: adversarial training, certified training using expressive losses and our proposed Feature Subset Certified Training.

5.4.1 Dichotomy search for irrelevant features

We implement the VeriX-incomplete algorithm as described in Algorithm 2, using Auto_LiRPA [46] as the underlying sound but not complete verifier. We also use a dichotomy search for the biggest irrelevant set following the improvement on VeriX from Wu et al. [176]. We describe the procedure in Algorithm 3. The VeriX-incomplete algorithm can then be used to finish the computation of the irrelevant set, starting with the output of the dichotomy search as $\mathcal{I}$ and perturbing the features of σ not yet proven irrelevant. We also consider the possibility of early stopping and only relying on the dichotomy search to find a subset of irrelevant features.
Similarly to the introduction of sound but incomplete verifier in the VeriX loop, doing early stopping still *gives a guaranteed lower bound on the size of the irrelevant set*, but does not guarantee that the set is maximal.
We notice experimentally that after the dichotomy search the irrelevant set is very close to the final maximal set.

Dichotomy search on MNIST with VeriX-complete We evaluate this using a complete version of VeriX on the first 100 samples of the MNIST dataset, analyzing a simple feedforward network, with a perturbation budget $\epsilon = 0.05$ and the same ordering σ as in Wu et al. [71][6]. The results are shown in Table 5.2. On average only 46 additional features are proven irrelevant after the initial dichotomy search when using a complete verifier. This represents 5.8% of the total number of features (784 for MNIST). When moving to an incomplete verifier[7], the importance of the first dichotomy search is even more pronounced as only 5 additional features are proven irrelevant on average after the initial dichotomy search.

Dichotomy search on CIFAR-10 with VeriX-incomplete. We also evaluate the difference of sizes of the irrelevant set after the dichotomy search and after the full execution when using an incomplete verifier on

```
DichotomySearch(x, ϵ, f_θ):
  σ ← TraversalOrder(f_θ, x) = (i_1, i_2, ..., i_n)
  y_pred ← argmax_k f_θ(x)[k]
  I_c ← σ[1 : ⌊n/2⌋]    # first candidate: first half of ordering
  I_f ← ∅                          # last successful irrelevant set
  σ_rem ← σ[⌊n/2⌋ + 1 : n] # remaining untested pixels

  while True do
      ϕ ← Perturb(x, I_c, ϵ)
      res ← Verify(ϕ, f_θ, y_pred)
      if res = True then
          I_f ← I_c
          # add half of the remaining unused pixels
          m ← |σ_rem|
          addSet ← σ_rem[1 : ⌊m/2⌋]
          if addSet = ∅ then break
          I_c ← I_c ∪ addSet
          σ_rem ← σ_rem[⌊m/2⌋ + 1 : m]
      else
          # remove half of current candidate
          k ← |I_c|
          remSet ← I_c[⌊k/2⌋ + 1 : k]
          if remSet = ∅ then break
          I_c ← I_c \ remSet
      end if

  end while

  I ← I_f

  return I
```

Algorithm 3: First steps of VeriX+: dichotomy search

Verifier	$\lvert\mathcal{I}\rvert$	Δ_{dicho}
Complete	619 ± 73	46 ± 58
α−CROWN	462 ± 156	5 ± 7

Table 5.2: Additional irrelevant features after the dichotomy search on MNIST. $\lvert\mathcal{I}\rvert$ is the mean size of the final irrelevant set after the full execution of VeriX-incomplete and Δ_{dicho} is the mean number of additional features proven irrelevant by the VeriX-incomplete algorithm after the dichotomy search.

CIFAR-10. We use a convolutional network and two training procedures: standard training (Clean) and certified training using the CC-IBP loss [8] (CC-IBP). We use $\epsilon = 4/255$ and the four traversal orders described in Section 5.3. For the underlying verifier we use CROWN. The results are shown in Table 5.3. The impact of the dichotomy search is even more pronounced than on MNIST as only a handful of additional features are proven irrelevant after the initial dichotomy search.
For the following experiments we will use only the dichotomy search as described in Algorithm 3 algorithm to compute a lower bound on the size of the irrelevant set.

5.4.2 Traversal orders comparison

Table 5.3 hints at a difference in performance between the traversal orders depending on the type of training used.
We further investigate this phenomenon on an adversarially trained model (Adv), a certified trained model (CC-IBP), a standard trained model (Standard) and a model trained with our Features Subset Certified Training with a CC-IBP loss (FS-CC-IBP). We use two datasets: CIFAR-10 and TinyImageNet. We here focus on the traversal order comparison. We refer to the next section for more detailed discussion of the training

TinyImageNet is a subset of ImageNet with 200 classes and 500 training samples and 50 test samples per class and images downsized to $64 \times 64 \times 3$.

Table 5.3: Additional irrelevant features after the dichotomy search on CIFAR-10. Evaluation results are grouped by model and traversal orders. The additional irrelevant features proven aftter the dichotomy search Δ_{dicho} are minimal, especially when using certified training. We highlight in bold the best traversal order for earch model (larger size of irrelevant set is better). On the clean model IBP bounds ordering performs best while on the CC-IBP model the linear coefficients ordering is the best.

Model	Traversal Order	$\lvert\mathcal{I}\rvert$	Δ_{dicho}
Clean	occlusion	18 ± 10	6 ± 5
	gradient	33 ± 16	6 ± 5
	ibp bounds	**40 ± 20**	3 ± 3
	crown coefs	27 ± 12	5 ± 4
CC-IBP	occlusion	163 ± 92	4 ± 3
	gradient	275 ± 150	2 ± 1
	ibp bounds	278 ± 157	2 ± 2
	crown coefs	**297 ± 152**	1 ± 1

procedures and the different trade-offs they offer.
The results are shown in Table 5.4. For all models except the standardly trained one, the linear coefficient ordering performs best, although the difference with IBP bounds or gradient ordering is not large. Interestingly, the IBP bounds ordering performs best for the standardly trained model. We hypothesize that the robustness induced by adversarial or certified training yields models with more meaningful gradients, making the saliency maps a better indicator of important features. Robustness also helps with the CROWN-coefficient heuristic as robust models have tighter intermediate bounds, introducing less approximation when linearizing the networks and yielding more meaningful coefficients.

Table 5.4: Comparison of traversal orders on CIFAR-10 and TinyImageNet using VeriX-incomplete with only the dichotomy search. Larger size of the irrelevant set is better. For each model we highlight in bold the best traversal order. The epsilon used is $\epsilon = 4/255$ for both datasets. The results are averaged over 300 samples of the test set for CIFAR-10 and 200 samples of the test set for TinyImageNet. For CIFAR-10 the total number of features is 1024 and for TinyImageNet it is 4096.

Dataset	Method	$\lvert\mathcal{I}\rvert$-Heuristic	$\lvert\mathcal{I}\rvert$-IBP Bounds	$\lvert\mathcal{I}\rvert$-CROWN Coefs	$\lvert\mathcal{I}\rvert$-Grad
CIFAR-10	Adv	89 ± 55	110 ± 72	**112 ± 74**	99 ± 69
	CC-IBP	178 ± 99	305 ± 154	**326 ± 154**	302 ± 151
	Standard	12 ± 10	**37 ± 22**	20 ± 11	25 ± 15
	FS-CC-IBP	67 ± 51	124 ± 96	**140 ± 93**	129 ± 92
TinyImageNet	Adv	228 ± 147	318 ± 218	**344 ± 228**	260 ± 200
	CC-IBP	416 ± 299	671 ± 475	**782 ± 498**	723 ± 489
	Standard	21 ± 19	**68 ± 46**	23 ± 23	39 ± 29
	FS-CC-IBP	185 ± 144	364 ± 394	**503 ± 429**	421 ± 396

For the rest of the experiments we will use the linear coefficient ordering as it performs best on robust models and is reasonably fast to compute.

5.4.3 Scalable formal explanations on CIFAR-10

We first justify the need for robust training by presenting experimental results obtained when lowering the perturbation budget on a standard model. We then compare three different training procedures to promote formal explainability: adversarial training, certified training using expressive losses and our proposed Feature Subset Certified Training.

Low perturbation budgets for scalable formal explanations We study as a preamble the simple strategy of lowering perturbation budgets to formally explain a standard network. We show in Table 5.5 the results of VeriX-incomplete with only the dichotomy search on a `CNN-7` architecture trained only on standard samples of CIFAR-10. Lowering the perturbation budget increases the size of the irrelevant set, but the empirical ϵ-formal explainability rate plunges as the perturbation budget is lowered. Even on a standard model, it is not possible to find counter examples for small perturbations. In fact, for the lowest perturbation radius $\epsilon = {}^{0.25}/_{255}$, among the 260 non empirically ϵ-formally explainable samples 33 have **provably** trivial explanations. Verification with CROWN is enough to show the model prediction cannot be changed within the perturbation budget on those samples.

ϵ-test	# $\text{EFX}^{\epsilon}_{AA}$	$\lvert\mathcal{I}\rvert$
${}^{0.25}/_{255}$	${}^{40}/_{300}$	747 ± 234
${}^{0.5}/_{255}$	${}^{82}/_{300}$	418 ± 160
${}^{1}/_{255}$	${}^{174}/_{300}$	210 ± 81

Table 5.5: CIFAR-10 results of VeriX-incomplete with only the dichotomy search and CROWN as the underlying verifier. The results are averaged over 300 samples of the test set. The total number of features is 1024. We report under $\text{EFX}^{\epsilon}_{AA}$ the number of empirically ϵ-formally explainable samples as defined in Definition 5.1.3, using AutoAttack as the adversary.

For the next experiments we increase the perturbation budgets, keeping two values to compare the impact of the perturbation budget on the trade-off between clean accuracy, empirically ϵ-formally explainable rate and size of the irrelevant set. In the following experiments we attempt to answer two questions:

- How simply lowering the perturbation budget with existing training procedure (adversarial training or certified training) impact the trade-off between clean accuracy, empirically ϵ-formally explainable rate and size of the irrelevant set?
- Can Feature Subset Certified Training further improve this trade-off?

We first focus on CIFAR-10 with two perturbation budgets for the computation of irrelevant features: $\epsilon = {}^{4}/_{255}$ and $\epsilon = {}^{8}/_{255}$. We chose relatively large perturbations budget, similar to the ones used in the literature for certified training on CIFAR-10 [8, 98, 104, 105], showcasing the scalability of our approach. For all the CIFAR-10 experiments we use the convolutional network with 7-layer `CNN-7`, commonly used in the certified robustness literature [8, 98, 104, 105].

Adversarial training setup We train three adversarially trained baseline with training perturbation budgets ϵ-train $\in \{{}^{1}/_{255}, {}^{4}/_{255}, {}^{8}/_{255}\}$. The goal is to evaluate the impact of adversarial training with a reduced perturbation budget to favor a good trade-off between clean accuracy, empirically ϵ-formally explainable rate and size of the irrelevant set.
For all models, we use the short schedule with cyclic learning rate of Chapter 4 described in Section 4.5. We employ a PGD attack [37] with 10

[37]: Madry et al. (2018), 'Towards Deep Learning Models Resistant to Adversarial Attacks'

iterations, a step size of $0.25 \times \epsilon$ train, SGD with weight decay of 5e-4 as the optimizer.

Certified training setup We also train three certified training baselines using the CC-IBP loss [8]. We reuse the same setting as the model tuned for robustness to $\epsilon = 2/255$ in De Palma et al. [8]: a schedule of 160 epochs with the training perturbation ramped up from 0 to the target ϵ for 80 epochs, the specialized initialization and regularization of Shi et al. [98] and L1 regularization with a coefficient of 3e-6. The learning rate is initialized at 5e-4 and decayed by a factor 5 at epochs 120 and 140, with Adam [199] as the optimizer.
We choose two different target perturbation budgets for training: $\epsilon \in \{0.5/255, 1/255\}$. In addition to the lower perturbation budgets we deviate from De Palma et al. [8] in two aspects[8]:

- a weaker attack: for the adversarial component of the loss we use the one step attack RS-FGSM [107] with the same target epsilon as the IBP component of the loss.
- lower ibp coefficient: we use α of 1e-3 in the weighted sum of the loss.

Feature Subset Certified Training setup For the FSCT models a perturbation budget of $\epsilon \in \{2/255, 4/255\}$ during training, dividing the target perturbation budget of VeriX by 2. We use the same schedule, hyperparameters and attack as the CC-IBP models. For the subset selection we use $\mu = 5\%$ and $\tau = 10$, perturbing 5% of the inputs from epoch 10 onwards for the model trained at $\epsilon = 2/255$ and $\mu = 20\%$, $\tau = 10$ for the model trained at $\epsilon = 4/255$ with $\mu = 20\%$.

Evaluation setup We evaluate the models using VeriX-incomplete with CROWN as the underlying verifier, the CROWN-coefficient traversal order, using only the dichotomy search to compute a lower bound on the size of the irrelevant set. We compute the empirically ϵ-formally explainable rate ($\text{EFX}^{\epsilon}_{AA}$) using AutoAttack [79] to find adversarial examples within the perturbation budget ϵ used for the computation of irrelevant features. Explanations are computed on the first 30 samples for each class of the test set, ensuring class balance for our evaluation on a total of 300 samples. The clean accuracy is evaluated on the entire test set. We report the results in Table 5.6.
As expected, all forms of robust training greatly improve the size of the irrelevant set $\mathcal{I}$, but they offer different trade-offs between clean accuracy, empirically ϵ-formally explainable rate and size of the irrelevant set.

Results on $\epsilon = 4/255$ In the lower perturbation regime, reducing the perturbation budget for adversarial training incurs a significant cost on the size of the irrelevant set, with a drop of more than 50% when moving from $\epsilon = 4/255$ to $\epsilon = 1/255$ (234 vs 112), while the number of empirically ϵ-formally explainable images remains low with 117 samples out of 300 not being empirically ϵ-formally explainable. Adversarial training results in too high overall empirical robustness, making it difficult to find adversarial examples within the perturbation budget for many samples.
Using a low perturbation budget with CC-IBP provides a better trade-off: the resulting verifiable robustness leads to larger irrelevant sets at a relatively low cost on clean accuracy. However, the rate of empirically ϵ-formally explainable, while better than adversarial training, remains low,

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

We recall the definition of the CC-IBP loss (Eq. (2.19)):

$$\mathcal{L}_{\alpha,CC-IBP} := \mathcal{L}\Big(- \big[(1-\alpha)\cdot z_{f_\theta}(x_{\text{adv}}, y) + \alpha \cdot \underline{z}_{f_\theta}(x, y)\big]; y\Big).$$

[199]: Kingma and Ba (2014), 'Adam: A method for stochastic optimization'

We use larger perturbation budgets for adversarial training as it is known to be much less effective at inducing verifiable robustness. This is also the reason we consider the setting with the same perturbation budget for training and computation of irrelevant features only for adversarial training.

8: In their work the attack is computed with PGD with 8 iterations. They also increase the perturbation used for the attack by a factor 2.1. The IBP coefficient α is set to 1e-2 after tuning.

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

We obtain these values with a grid search with $\mu \in \{5\%, 10\%, 20\%\}$ and $\tau \in \{5, 10, 20\}$.

[79]: Croce and Hein (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'

ϵ-test	Method (ϵ-train)	Accuracy (%)	# $\text{EFX}^{\epsilon}_{AA}$	$\lvert\mathcal{I}\rvert$
$\frac{4}{255}$	Standard	90.31	$^{299}/_{300}$	20 ± 11
	PGD ($^{1}/_{255}$)	88.82	$^{183}/_{300}$	112 ± 74
	PGD ($^{4}/_{255}$)	82.67	$^{120}/_{300}$	234 ± 149
	CC-IBP ($^{0.25}/_{255}$)	88.14	$^{257}/_{300}$	241 ± 120
	CC-IBP ($^{0.5}/_{255}$)	87.19	$^{224}/_{300}$	**326 ± 154**
	FS-CC-IBP ($^{2}/_{255}$)	**88.87**	**$^{294}/_{300}$**	140 ± 93
$\frac{8}{255}$	Standard	90.31	$^{293}/_{300}$	7 ± 4
	PGD ($^{4}/_{255}$)	82.67	$^{212}/_{300}$	84 ± 74
	PGD ($^{8}/_{255}$)	73.89	$^{178}/_{300}$	130 ± 117
	CC-IBP ($^{0.5}/_{255}$)	87.19	**$^{299}/_{300}$**	134 ± 88
	CC-IBP ($^{1}/_{255}$)	85.57	$^{291}/_{300}$	**200 ± 114**
	FS-CC-IBP ($^{4}/_{255}$)	**87.64**	**$^{299}/_{300}$**	171 ± 95

Table 5.6: Evaluation results on CIFAR-10. Larger size of the irrelevant set is better. The total number of features for CIFAR-10 is 1024. We highlight in bold the best result for each metric and for each ϵ-test, focusing on the robust training methods. We include the standardly trained model to gage the trade-offs. Expectedly, it offers the best clean accuracy, but not necessarily the best empirically ϵ-formally explainable rate: for some samples it is not possible to find even one robust feature.

with 33 samples out of 300 not being empirically ϵ-formally explainable for the model trained with the lowest perturbation budget ($\epsilon = {}^{0.25}/_{255}$). The use of FSCT with CC-IBP (FS-CC-IBP) provides an interesting trade-off: it has the highest rate of empirically ϵ-formally explainable (294 out of 300) and the highest clean accuracy among all the robust models, with a larger average irrelevant set than the the adversarial model trained at $\epsilon = {}^{1}/_{255}$ (140 vs 112).
Overall on this setting it is challenging to obtain a large irrelevant set while maintaining a high rate of empirically ϵ-formally explainable: inducing even a little robustness very quickly leads to the impossibility of finding adversarial examples within the perturbation budget for some of the samples. While FS-CC-IBP is better at mitigating this issue, it is not completely solved and there is a cost on the size of the irrelevant compared to CC-IBP.

Results on $\epsilon = {}^{8}/_{255}$ On the higher perturbation regime ($\epsilon = {}^{8}/_{255}$) adversarial training is even less effective. The robustness still entails the largest number of samples that are not empirically ϵ-formally explainable (89 out of 300 for ϵ-train = ${}^{4}/_{255}$ and 117 out of 300 for ϵ-train = ${}^{4}/_{255}$) among all the robust training methods. Both CC-IBP and FS-CC-IBP exhibit a larger number of irrelevant features while having a much lower number of samples that are not empirically ϵ-formally explainable. FS-CC-IBP here offers a good trade-off between the two CC-IBP models, with larger irrelevant sets than the model trained at $\epsilon = {}^{0.5}/_{255}$ while maintaining comparable rate of empirically ϵ-formally explainable and clean accuracy. Increasing the perturbation budget with CC-IBP leads to the largest irrelevant set but at the cost of a lower clean accuracy and a lower rate of empirically ϵ-formally explainable.

5.4.4 Scalable formal explanations on TinyImageNet

We use a similar study on TinyImageNet to further show the scalability of our approach. TinyImageNet is a subset of ImageNet [21], using 200 classes out of the original 1000 and images of size 64x64x3.

[21]: Deng et al. (2009), 'Imagenet: A large-scale hierarchical image database'

General training setup We make a notable change in the architecture of the `CNN-7` used in the literature: we replace the flattening and linear layer directly after the last convolutional layer with a global average pooling. This change is necessary to reduce overfitting for the standard baseline, a particular concern on TinyImageNet.[9]
We use a schedule of 70 epochs with an initial learning rate of 5e-4, decayed by a factor 5 at epochs 50 and 60, with Adam as the optimizer, for all the training methods.
We consider training perturbation budgets ϵ-train in $\{1/255, 4/255, 8/255\}$ for the adversarially trained models. The attack is also computed with PGD with 10 iterations and a step size of $0.25 \times \epsilon$-train.
The CC-IBP models are trained with perturbation budgets ϵ-train $\in \{0.25/255, 0.5/255, 1/255\}$, with RS-FGSM [6] as the attack for the adversarial component of the loss and an IBP coefficient α of 1e-3.
Finally, we use the same parameters for the FS-CC-IBP models as for CIFAR-10, with ϵ-train $\in \{2/255, 4/255\}$ and subset selection parameters $\mu = 5\%$, $\tau = 10$ for the model trained at $\epsilon = 2/255$ and $\mu = 20\%$, $\tau = 10$ for the model trained at $\epsilon = 4/255$.[10] The perturbation budget is ramped up from 0 to the target ϵ for the initial 20 epochs for CC-IBP and FS-CC-IBP models.

9: This is due notably to the relatively low number of samples per class (500 samples for each class in the training set). The original version of the `CNN-7` architecture is popular in the certified training literature even for TinyImagenet, *e.g.* in De Palma et al. [8] or Mao et al. [174] since standard training is not used as a baseline.

[6]: Wong et al. (2020), 'Fast is better than free: Revisiting adversarial training'

10: We did a grid search with the same values as for CIFAR-10, and found the same final parameters to provide good results.

Evaluation setup We use the same evaluation setup as for CIFAR-10, with VeriX-incomplete using CROWN as the underlying verifier, the CROWN-coefficient traversal order and only the dichotomy search to compute a lower bound on the size of the irrelevant set. We compute the empirically ϵ-formally explainable rate ($\text{EFX}^{\epsilon}_{AA}$) using AutoAttack to find adversarial examples within the perturbation budget ϵ used for the computation of irrelevant features. Explanations are computed on the first 5 samples of each class of the test set (1000 samples in total) and the clean accuracy is evaluated on the entire test set. The results are reported in Table 5.7.

Table 5.7: Evaluation results on TinyImageNet. The total number of features is 4096 (64×64). We highlight in bold the best result for each metric and for each ϵ-test, focusing on the robust training methods. We include the standardly trained model to gage the trade-offs. Expectedly, it offers the best clean accuracy and empirically ϵ-formally explainable rate as it is always possible to find adversarial examples within the perturbation budget. However, it offers the worst size of the irrelevant set.

ϵ-test	Method (ϵ-train)	Accuracy (%)	# $\text{EFX}^{\epsilon}_{AA}$	$\lvert\mathcal{I}\rvert$
$\frac{4}{255}$	Standard	54.80	$990/1000$	23 ± 27
	PGD ($1/255$)	**52.61**	$911/1000$	340 ± 217
	PGD ($4/255$)	41.41	$763/1000$	741 ± 496
	CC-IBP ($0.25/255$)	46.20	$976/1000$	558 ± 378
	CC-IBP ($0.5/255$)	44.79	$947/1000$	**804 ± 514**
	FS-CC-IBP ($2/255$)	48.60	**$996/1000$**	506 ± 393
$\frac{8}{255}$	Standard	54.80	$887/1000$	6 ± 7
	PGD ($4/255$)	41.41	$925/1000$	263 ± 242
	PGD ($8/255$)	29.32	$882/1000$	617 ± 485
	CC-IBP ($0.5/255$)	**44.79**	$997/1000$	306 ± 227
	CC-IBP ($1/255$)	42.36	$988/1000$	465 ± 323
	FS-CC-IBP ($4/255$)	44.49	**$998/1000$**	**684 ± 484**

Results on TinyImageNet Similarly to CIFAR-10 the standard model does not have every sample as empirically ϵ-formally explainable: for

some samples it is not possible to find even one robust feature among the 4096 input features. This highlights the need for some robustness to obtain meaningful explanations.
At ϵ-test = $4/255$, the FS-CC-IBP model has a more interesting trade-off than for CIFAR-10. It has the highest rate of empirically ϵ-formally explainable (996 out of 1000) and the average number of irrelevant features is about 50% larger than the adversarial model trained at $\epsilon = 1/255$ (506 vs 340), and only 10% below the CC-IBP model trained with the lowest perturbation budget. However, the cost on the clean accuracy is higher, with a drop of about 6 percentage points from the standard model.
We note that the robust baselines, in particular the CC-IBP models, have a higher rate of empirically ϵ-formally explainable on TinyImageNet than on CIFAR-10. This is likely due to the challenge of the dataset (higher number of classes but fewer samples per class, more complex images) making it easier to find adversarial examples.
At ϵ-test = $8/255$ FS-CC-IBP provides the best trade-off, with the highest rate of empirically ϵ-formally explainable and the largest average irrelevant sets. Its clean accuracy is only slightly below the CC-IBP model at ϵ-train = $0.5/255$, which has the best clean accuracy over the robust models. We note that this CC-IBP model also has comparable rate of empirically ϵ-formally explainable but a much lower average number of irrelevant features (306 vs 684). While it can be increased by training with a larger ϵ, doubling it is not enough to bridge the gap, with a lower clean accuracy, lower rate of empirically ϵ-formally explainable and still a smaller irrelevant set on average.
For both ϵ-test, adversarial training has the worst rate of empirically ϵ-formally explainable overall, and decreasing the perturbation budget is not enough to reach the explainability performance of CC-IBP or FS-CC-IBP.
We note that adversarial training at ϵ-train = $8/255$ is particularly challenging, with a large cost on clean accuracy (29.32%). Further efforts to find better training hyperparameters for this setting should improve the trade-off.

5.5 Related work

We first discuss improvements to the original VeriX algorithm [71] focused on reducing the number of calls to the verifier without trading off minimality of the explanations. We then present approaches that trade off minimality for scalability, either by using incomplete verifiers or other heuristics.

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

Improving formal explainability without trading off minimality VeriX + [176], building on VeriX [71], proposes several improvements: a new ordering, IBP bounds, discussed in Section 5.3.1, the dichotomy search described in Algorithm 3, with an additional optimization specific to their implementation of the verifier calls. All these improvements do not trade off minimality of the explanations but rather reduce the number of calls to the verifier in practice.

[176]: Wu et al. (2024), 'Better Verified Explanations with Applications to Incorrectness and Out-of-Distribution Detection'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

Using incomplete verifiers Wu et al. [71] uses DeepPoly [40] as an incomplete verifier and show the trade-off between the size of the irrelevant set and the computation time on standardly trained model with small perturbation budgets ($\epsilon = 0.05$ for MNIST and $\epsilon = 0.005$ for GTRSB).[11].
Doncenco et al. [177] uses PyRAT [59] (with a zonotope abstraction) and

[40]: Singh et al. (2019), 'An Abstract Domain for Certifying Neural Networks'

11: GTRSB is the German Traffic Sign Benchmark [200]. It contains only images of digits but in color and with dimension 32×32.

[177]: Doncenco et al. (2025), 'A Dive into Formal Explainable Attributions for Image Classification'

[59]: Lemesle et al. (2024), 'Neural Network Verification with PyRAT'

CROWN-IBP [107] as incomplete verifiers, together with Marabou. Their idea is to treat feature attributions as traversal order heuristics in VeriX, and then evaluate them by measuring the sizes of the resulting formal explanations. In this setup, smaller explanations correspond to better attribution methods. They show that although the use of incomplete verifiers introduces some loss of minimality, this does not affect the relative ranking of attribution methods.

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

Robust training and formal explainability There is little discussion in the literature on the impact of the training method on the explainability of the resulting models. Wu et al. [176] consider a form of adversarial training[12] on a MNIST Fully Connected network. They show that the explanation size is reduced when the model is adversarially trained. The model has poor formal explainability: it is certifiably robust on more than 50% of the samples. Doncenco et al. [177] evaluate models trained with certified training at different perturbations levels. They do not report their formal explainability in terms of ϵ-formally explainable rate (empirical or verified). They focus on their metric comparing the attribution methods used as traversal order: it remains consistent across perturbation levels.

12: More specifically, they randomly compute adversarial perturbation on half of the training set. The perturbation are computed on a pretrained model. The final model is thus trained on a set containing a mix of clean samples and attacked samples transferred from another model.

Other approaches trading off minimality for scalability Several works have introduced approaches to compute explanations of larger sizes, losing the guarantee of minimality for a more efficient computation. Both Wu et al. [176] and Izza et al. [178] adapted the QuickXplain algorithm Junker [201] to the formal explainability setting. QuickXplain is a divide-and-conquer algorithm allowing parallelization of the computation of the explanations but losing the guarantee of minimality.
Doncenco et al. [177] and Bassan and Katz [72] propose to consider subsets of features instead of single features. They first partition the input into **super-pixels** (called **bundles** in Bassan and Katz [72]), then apply VeriX or a similar approach to find a minimal set of super-pixels that must be fixed to maintain the classification. This approach can be parametrized by the size and number of super-pixels, trading off the granularity of the explanations for scalability. The explanations are not minimal since a super-pixel can contain both relevant and irrelevant features.[13] Note that Bassan and Katz [72] do not consider ϵ-robust explanations but formal explanations with no restriction on the perturbation budget.
Finally, Bassan et al. [202] build on an abstraction-refinement framework for neural network verification [203]. The neurons of the network are merged to build an abstracted network, allowing faster verification. It is also a trade-off between performance and computation time. By construction the explanation of the abstracted network is a superset of the explanation of the original network. The abstraction can be gradually refined to obtain smaller explanations.
These approaches are orthogonal to our work: they can be used in conjunction with our training method to further improve scalability. We find it promising future work to evaluate the combination of our training method with some of these approaches.

[178]: Izza et al. (2024), 'Distance-restricted explanations: theoretical underpinnings & efficient implementation'

[201]: Junker (2004), 'QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems'

[72]: Bassan and Katz (2023), 'Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks'

13: The super-pixels proven irrelevant only contains irrelevant features but the super-pixels remaining unknown or proven relevant can contain both relevant and irrelevant features even if the underlying verifier is complete.

[202]: Bassan et al. (2025), 'Explaining, Fast and Slow: Abstraction and Refinement of Provable Explanations'

[203]: Ladner and Althoff (2025), 'Fully Automatic Neural Network Reduction for Formal Verification'

Training for explainability Sufficient Subset Training (SST), proposed in Bassan et al. [204], is closely related to our work. They propose a dual-propagation training method to train self-explainable models with conciseness as one of the goals. In addition to training standardly on clean samples, SST involves learning to mask and maintain prediction on masked samples. This second task involves two stages: producing a mask, and classifying samples perturbed according to the mask. To achieve this they modify a standard convolutional network[14] so that it outputs two

[204]: Bassan et al. (2025), 'Explain Yourself, Briefly! Self-Explaining Neural Networks with Concise Sufficient Reasons'

14: They use a ResNet18.

vectors: the usual logit vector, and a vector of scores in $[0,1]^d$, with d the dimension of the samples considered. Those scores are produced by adding a decoder to the network with a sigmoid activation as a last layer. Features with a score above a threshold are unchanged, and the others are perturbed. The masks are then used to select the subset to perturb. Three different strategies are considered: using random noise, replacing with a baseline value (typically 0) or using an adversarial perturbation. The perturbed inputs are used to train the network, using the predicted label of the clean sample as the target.
Our work differs in several ways. First, we remain in the framework of formal explanations. Our training method is designed to help the computation of formal explanations. The irrelevant features we considered are provably irrelevant, even if the irrelevant set is not maximal. Consequently, we rely on over-approximations during training to enforce verifiable robustness on subset of features, rather than only relying on adversarial attacks as done with the robust masking strategy in Sufficient Subset Training (for both training and evaluation). Secondly, we do not claim to train **self-explainable** models. In SST, the binary mask obtained with the threshold is considered as the explanation of the model. There is no notion of formal explanation anymore: the mask is evaluated post-training using the same perturbation heuristics as used during training. It is also compared only to heuristic post-hoc explanation methods [205–207].

[205]: Fong and Vedaldi (2017), 'Interpretable Explanations of Black Boxes by Meaningful Perturbation'

[206]: Ribeiro et al. (2018), 'Anchors: high-precision model-agnostic explanations'

[207]: Carter et al. (2019), 'What made you do this? Understanding black-box decisions with sufficient input subsets'

Over-approximations for feature attribution The work of Fel et al. [208] also uses bounding methods to compute explanations. However, they do not aim to find formal explanations in the sense of Wu et al. [71], but rather to compute feature attributions, scoring the features according to the difference in the output bounds when the feature is perturbed or not. Given output bounds of the network, they compute the **adversarial overlap**: the lower bound of the logit of the predicted class minus the upper bound of the logit of the second-highest class. The adversarial overlap is computed when every feature is perturbed, then when the feature in a subset of interest are fixed. The drop in adversarial overlap is used as a score for the features in the subset. The larger the drop, the more important the features in the subset are considered to be. It can be seen as a more involved version of the bounding heuristic for the traversal order of VeriX-incomplete described in Section 5.3.1, and it would be interesting future work to evaluate it as a traversal order heuristic.

[208]: Fel et al. (2023), 'Don't Lie to Me! Robust and Efficient Explainability with Verified Perturbation Analysis'

5.6 Conclusion and future work

In this chapter, we have taken first steps toward training for formal explainability by training models under partial perturbations restricted to feature subsets with Feature Subset Certified Training (FSCT). We also discussed the notion of empirically ϵ-formally explainable, a crucial aspect to compare model's formal explainability beyond the size of the explanations. "Our new traversal order, inspired by the heuristic described in Chapter 3, consistently outperforms existing heuristics for all robust models we trained. Preliminary results indicate that FSCT is an interesting trade-off between explanation size and the rate of empirically ϵ-formally explainable compared to other robust training methods.
We plan to further investigate the applicability of FSCT in combination with different certified losses (*e.g.*, MTL-IBP [8], CROWN-IBP [107]) and alternative architectures such as ResNets [75]. A more ambitious direction is to frame feature masking as a learnable component, leveraging

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[107]: Zhang et al. (2020), 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'

[75]: He et al. (2016), 'Identity mappings in deep residual networks'

segmentation techniques to determine which—and how many—features should be masked in a full end-to-end training.
The greatest challenge is to train models where incomplete methods provide both partial verification (*e.g.*, via over-approximations such as CROWN, as done in this work) and falsification (*e.g.*, adversarial attacks limited to robust features plus one unknown feature).

Conclusion

6 Conclusion and Perspectives

Contributions

In this thesis, we studied how bound propagation and linear approximations of neural networks can be leveraged to improve verification, empirical robustness, and formal explainability. Verification is an important problem but is known to be NP-complete [134]. Making verification tractable is an active area of research. In a first contribution we study in particular input partitioning, a form of branch-and-bound. Due to this tractability issue, the field of certified training has emerged as a way to incorporate verifiability in the training process. Certified training heavily relies on bound propagation, often using the loose but scalable interval bound propagation (IBP) method. Certified training is well studied in the context of provable robustness. In this thesis, we propose to study it in two other applications: improving empirical robustness and formal explainability.

Tightening Bounds for Incomplete Verification The field of neural network verification is very active and there are many works studying branch-and-bound, a key ingredient of most neural network verifiers. For networks with high-dimensional input spaces, the focus has been on branching on internal nodes of the networks. Input partitioning is another form of branching, simple yet effective in some settings. This approach is particularly interesting for networks with low-dimensional inputs where the input ranges specified by the property of interest are large. We proposed ReCIPH in Chapter 3, a heuristic to speedup branching in the case of input partitioning. Its goal is to reduce the number of subproblems to be analyzed. ReCIPH leverages the linear coefficients readily available from bound propagation methods to rank inputs for partitioning. It supports any network supported by the bound propagation method, and is easily parallelized. ReCIPH is implemented in PyRAT [1]. We showed that ReCIPH significantly outperforms the largest-interval strategy on several benchmarks. The number of subproblems to analyze is comparable to that of the gradient smear heuristic [58], but the latter requires additional computation, making ReCIPH more advantageous. PyRAT, employed with ReCIPH on several benchmarks, got competitive results from a recent edition of the International Verification of Neural Networks Competition (VNN-COMP 2024) [60].

1: https://pyrat-analyzer.com/

[58]: Wang et al. (2018), 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'

[60]: Brix et al. (2024), 'The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results'

Certified Training for Empirical Robustness In Chapter 4 we proposed to use a recent framework for certified training, Expressive Losses [8], to improve empirical robustness. Multi-step adversarial training, a popular method for improving empirical robustness, suffers from computational overhead because it requires generating adversarial examples using several forward and backward passes for a single batch during training. Single-step adversarial training was proposed to reduce this overhead but is known to suffer from catastrophic overfitting. We showed that in this context, expressive losses can be used to train empirically robust models without suffering from catastrophic overfitting, even in some settings where the popular randomized method N-FGSM [4] fails. We then studied the gap between the empirical robustness achieved by certified training and that achieved by multi-step adversarial training, and showed that

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[4]: Jorge et al. (2022), 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'

when tuned properly, expressive losses can reach performance similar to multi-step adversarial training in some settings.

Certified Training for Formal Explainability Robustness is a highly desirable property, especially in the context of safety-critical applications. However, trustworthiness of a model can also rely on its explainability. Formal Explainability [53, 71, 72, 196] is an emerging field that connects provable robustness and explainability. This entails running a neural network verifier multiple times, considering features in a given order, to explain the network's decision on a given input: input features are deemed irrelevant when perturbing them does not change the network's decision. A good heuristic for the ordering of the features leads to concise explanations. There are active research efforts to make formal explainability tractable. In Chapter 5, we proposed using certified training for formal explainability in combination with an incomplete verifier to compute explanations. Fully robust networks can have void explanations, as their decisions remain the same for any input feature perturbation. Introducing incomplete verifiers requires a new notion for evaluating explanations, as some features can be in an unknown state, and the true explanation may be void if the unknown features are irrelevant. We introduce the notion of empirically ϵ-formally explainable networks: using an adversarial attack oracle, we can check if perturbing all features can fool the network, guaranteeing that the true explanation is not void. We introduce a new training method, Feature Subset Certified Training, that focuses on training for partial robustness to obtain formally explainable models. We evaluate our approach on medium-sized neural networks on two image datasets and find that reducing the perturbation radius and IBP loss weight in either adversarial training or certified training with an expressive loss is not always sufficient, and that Feature Subset Certified Training is a step toward training empirically ϵ-formally explainable networks. We also show that the ReCIPH heuristic can be employed for feature ordering, leading to more concise explanations when applied to robustly trained models

[53]: Marques-Silva and Ignatiev (2022), 'Delivering Trustworthy AI through Formal XAI'

[71]: Wu et al. (2023), 'VeriX: Towards Verified Explainability of Deep Neural Networks'

[72]: Bassan and Katz (2023), 'Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks'

[196]: La Malfa et al. (2021), 'On Guaranteed Optimal Robust Explanations for NLP Models'

Perspectives and future directions

Input Partitioning Single input bisection is not the only option for input partitioning. Choosing not only the input to split, but also how many other inputs to jointly split, where, and in how many subproblems at once, are all possibilities for a complex splitting strategy. A successful approach should be adaptative and generalizable to various networks and properties. An interesting direction is to frame it as a learning problem, but it is not straightforward. Starting with the dataset: what would constitute a sample in a supervised learning problem? The input could be the network and property, and the label the optimal partitioning strategy. Building such a dataset would not be tractable. The model taking the decision would also need to be carefully designed, to be able to handle various network architectures. Its output should not only be scores ranking the inputs but should also determine how many partitions per input are preferable and/or where to split. A starting point could be the Graph Neural Network approach to learning internal node branching proposed in [149], possibly combined with reinforcement learning, using the size of the partitioning tree as a penalty. It is not clear how beneficial an improved partitioning strategy would be: it remains applicable only to networks with low-dimensional inputs. Branching on internal nodes, a widely studied approach, is also applicable to such

[149]: Lu and Kumar (2020), 'Neural Network Branching for Neural Network Verification'

networks. However, as input partitioning is also used in combination with internal node branching, an improved input partitioning strategy could still be beneficial.

Future of Verification On a broader scope, many challenges remain in verification. Scalability is, of course, a never-ending challenge. Large Language Models (LLMs) exemplify the ever-growing size of neural networks. Beyond the scale issue they also introduce new challenges in the verification process. The attention mechanism of transformers [209], at the core of LLMs, requires handling dot product between variables, and linear approximations are ill-suited to handle such operations. Adapting approximations to transformers is an active area of research [210–212]. Certified training of transformers remains limited [46]. We do not foresee any major step toward formal verification of LLMs in the near future. However, they provide an important avenue for classical software verification: analysis of LLM generated code.
Another challenge is the specification problem: local robustness is well-defined but limited in scope. Properties involving several executions of the network, such as fairness, can be reduced to local robustness using self-composition [213, 214]. However, this reduction is not necessarily the most efficient approach. Dedicated solvers [49, 215] are generally more effective. Global robustness, defined as the robustness of a network on its data manifold, is a highly desirable yet elusive goal: the manifold is unknown. Using generative models to approximate the manifold and integrating it them the verification pipeline is an interesting direction [56].

Improving Feature Subset Certified Training Besides immediate extensions of FSCT (*e.g.* to other datasets and architectures), several improvements can be envisioned. The selection of features to be considered irrelevant at training time could be learned, and not heuristically chosen. Careful design and tuning of the architecture and loss are needed to learn good masking and avoid sacrificing accuracy. A major challenge in scaling formal explainability using incomplete methods is the difficulty of finding counterexamples, as the model has some robustness and the perturbation space is reduced. Two directions can be explored: encouraging some features to be relevant at training time (incorporated into the FSCT framework) and improving the counterexample search itself.

Connections Between Robustness and Explainability Explainability is highly related to neural network robustness. Provable robustness can be at odds with formal explainability, as discussed in Chapter 5. However, empirical robustness has been shown to have beneficial properties for heuristic-based explainability. On the one hand, adversarial robustness seems to produce more meaningful explanations [216–219]. On the other hand, training for explainability can improve local robustness [220–223]. This phenomenon is due to the better alignment of the input gradients of robust models with human perception. Certified training does not lead to such alignment: the regularization is too strong, and the input gradients become flat, producing saliency maps that are very difficult to interpret [224]. Interestingly, Lipschitz networks on the other hand have interesting properties toward explainability [225]. A possible direction to explore is examining the explainability properties of models trained with Expressive Losses [8] when tuned for this purpose, following our study in Chapter 4. Quantifying the quality of saliency maps can be done using the perceptual alignment metric introduced in [219]. It would also be interesting to investigate whether Feature Subset Certified Training,

[209]: Vaswani et al. (2017), 'Attention is all you need'

[210]: Bonaert et al. (2021), 'Fast and precise certification of transformers'

[211]: Munakata et al. (2022), 'Verifying Attention Robustness of Deep Neural Networks against Semantic Perturbations'

[212]: Shao et al. (2023), 'STR-Cert: Robustness Certification for Deep Text Recognition on Deep Learning Pipelines and Vision Transformers'

[46]: Xu et al. (2020), 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'

[213]: Athavale et al. (2024), 'Verifying Global Two-Safety Properties in Neural Networks with Confidence'

[214]: Boetius and Leue (2023), *Verifying Global Neural Network Specifications using Hyperproperties*

[49]: Urban and Miné (2021), 'A Review of Formal Methods applied to Machine Learning'

[215]: Banerjee et al. (2024), 'Input-Relational Verification of Deep Neural Networks'

[56]: Wu et al. (2022), 'Toward Certified Robustness Against Real-World Distribution Shifts'

[216]: Ilyas et al. (2019), 'Adversarial examples are not bugs, they are features'

[217]: Etmann et al. (2019), 'On the Connection Between Adversarial Robustness and Saliency Map Interpretability'

[218]: Chalasani et al. (2020), 'Concise Explanations of Neural Networks using Adversarial Training'

[219]: Srinivas et al. (2023), 'Which Models have Perceptually-Aligned Gradients? An Explanation via Off-Manifold Robustness'

[220]: Ross and Doshi-Velez (2017), 'Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients'

[221]: Boopathy et al. (2020), 'Proper Network Interpretability Helps Adversarial Robustness in Classification'

[222]: Ganz et al. (2023), 'Do Perceptually Aligned Gradients Imply Robustness?'

[223]: Joo et al. (2023), 'Towards More Robust Interpretation via Local Gradient Alignment'

[224]: Banerjee et al. (2024), 'Interpreting Robustness Proofs of Deep Neural Networks'

[225]: Serrurier et al. (2022), 'On the explainable properties of 1-Lipschitz Neural Networks: An Optimal Transport Perspective'

[8]: De Palma et al. (2024), 'Expressive Losses for Verified Robustness via Convex Combinations'

[219]: Srinivas et al. (2023), 'Which Models have Perceptually-Aligned Gradients? An Explanation via Off-Manifold Robustness'

introduced in Chapter 5, has any beneficial effect on heuristic-based explainability.

Bibliography

Here are the references in citation order.

[1] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 'Explaining and Harnessing Adversarial Examples'. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cited on pages vii, 4, 15, 16).

[2] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 'Semidefinite relaxations for certifying robustness to adversarial examples'. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 10900–10910 (cited on pages viii, 42).

[3] Ben Batten, Panagiotis Kouvaros, Alessio Lomuscio, and Yang Zheng. 'Efficient Neural Network Verification via Layer-based Semidefinite Relaxations and Linear Cuts'. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 2184–2190. DOI: 10.24963/ijcai.2021/301 (cited on pages viii, 42).

[4] Pau de Jorge, Adel Bibi, Riccardo Volpi, Amartya Sanyal, Philip Torr, Grégory Rogez, and Puneet K. Dokania. 'Make Some Noise: Reliable and Efficient Single-Step Adversarial Training'. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022 (cited on pages viii, ix, 16, 17, 47, 48, 50–53, 55, 61, 69, 95).

[5] Maksym Andriushchenko and Nicolas Flammarion. 'Understanding and Improving Fast Adversarial Training'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 16048–16059 (cited on pages viii, 17, 45–47, 51, 52, 69).

[6] Eric Wong, Leslie Rice, and J. Zico Kolter. 'Fast is better than free: Revisiting adversarial training'. In: *International Conference on Learning Representations*. 2020 (cited on pages viii, 5, 16, 47, 51, 52, 70, 88).

[7] Elias Abad Rocamora, Fanghui Liu, Grigorios Chrysos, Pablo M. Olmos, and Volkan Cevher. 'Efficient local linearity regularization to overcome catastrophic overfitting'. In: *The Twelfth International Conference on Learning Representations*. 2024 (cited on pages viii, 17, 46, 50–56, 69).

[8] Alessandro De Palma, Rudy Bunel, Krishnamurthy Dvijotham, M. Pawan Kumar, Robert Stanforth, and Alessio Lomuscio. 'Expressive Losses for Verified Robustness via Convex Combinations'. In: *The Twelfth International Conference on Learning Representations*. 2024 (cited on pages x, 5, 20, 21, 31, 45, 46, 50, 51, 58, 59, 70, 71, 73, 74, 78, 82, 83, 85, 86, 88, 91, 95, 97, 98).

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012 (cited on pages 1, 13).

[10] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 'Mastering the game of Go with deep neural networks and tree search'. In: *Nature* 529.7587 (Jan. 1, 2016), pp. 484–489. DOI: 10.1038/nature16961 (cited on page 1).

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, et al. 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901 (cited on page 1).

[12] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, et al. 'Highly accurate protein structure prediction with AlphaFold'. In: *Nature* 596.7873 (Aug. 1, 2021), pp. 583–589. DOI: 10.1038/s41586-021-03819-2 (cited on page 1).

[13] Frank Rosenblatt. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65 6 (1958), pp. 386–408 (cited on page 1).

[14] Seppo Linnainmaa. 'Taylor expansion of the accumulated rounding error'. In: *BIT Numerical Mathematics* 16.2 (June 1, 1976), pp. 146–160. DOI: 10.1007/BF01931367 (cited on pages 1, 14).

[15] Paul J. Werbos. 'Applications of advances in nonlinear sensitivity analysis'. In: *System Modeling and Optimization*. Ed. by R. F. Drenick and F. Kozin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 762–770 (cited on page 1).

[16] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323 (1986), pp. 533–536 (cited on page 1).

[17] Kunihiko Fukushima. 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36.4 (Apr. 1, 1980), pp. 193–202. DOI: 10.1007/BF00344251 (cited on pages 1, 14, 15).

[18] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. 'Parallel distributed processing model with local space-invariant interconnections and its optical architecture'. In: *Appl. Opt.* 29.32 (Nov. 1990), pp. 4790–4797. DOI: 10.1364/AO.29.004790 (cited on pages 1, 2, 15).

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 'Backpropagation Applied to Handwritten Zip Code Recognition'. In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541 (cited on pages 1, 2, 15).

[20] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 'Large-scale deep unsupervised learning using graphics processors'. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 873–880. DOI: 10.1145/1553374.1553486 (cited on pages 1, 2).

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 'Imagenet: A large-scale hierarchical image database'. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cited on pages 1, 2, 87).

[22] Jingyuan Zhao, Yuyan Wu, Rui Deng, Susu Xu, Jinpeng Gao, and Andrew Burke. 'A Survey of Autonomous Driving from a Deep Learning Perspective'. In: *ACM Comput. Surv.* 57.10 (May 2025). DOI: 10.1145/3729420 (cited on pages 1, 2).

[23] Md Manjurul Ahsan, Shahana Akter Luna, and Zahed Siddique. 'Machine-learning-based disease diagnosis: A comprehensive review'. In: *Healthcare*. Vol. 10. 3. MDPI. 2022, p. 541 (cited on pages 1, 2).

[24] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks*. ProPublica. Accessed: YYYY-MM-DD. May 2016. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (cited on pages 1, 2).

[25] Wikipedia contributors. *List of Tesla Autopilot crashes*. https://en.wikipedia.org/wiki/List_of_Tesla_Autopilot_crashes. Last major update Oct 2024; accessed: 2025-10-02. Oct. 2024 (cited on pages 1, 2).

[26] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 'Defending against neural fake news'. In: *Advances in neural information processing systems* 32 (2019) (cited on pages 1, 2).

[27] European Union. *Regulation (EU) 2024/1689 Artificial Intelligence Act*. https://eur-lex.europa.eu/eli/reg/2024/1689/oj. Official Journal of the European Union, L 202, 12 July 2024, p. 1–132. 2024. (Visited on 10/03/2025) (cited on pages 1, 2).

[28] Florent Kirchner, Nikolai Kosmatov, Virgile Prévosto, Julien Signoles, and Boris Yakobowski. 'Frama-C: A software analysis perspective'. In: *Formal Aspects of Computing* 27.3 (May 2015), pp. 573–609. DOI: 10.1007/s00165-014-0326-7 (cited on page 2).

[29] Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. 'Design and Implementation of a Special-Purpose Static Program Analyzer for Safety-Critical Real-Time Embedded Software'. In: *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*. Ed. by Mogensen, T., Schmidt, D.A., Sudborough, and I.H. Vol. 2566. Lecture Notes in Computer Science. Springer, 2002, pp. 85–108. DOI: 10.1007/3-540-36377-7_5 (cited on page 2).

[30] Edmund M. Clarke and E. Allen Emerson. 'Design and synthesis of synchronization skeletons using branching time temporal logic'. In: *Logics of Programs*. Ed. by Dexter Kozen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 52–71 (cited on page 2).

[31] T. Coquand and Gérard Huet. *The calculus of constructions*. Tech. rep. RR-0530. INRIA, May 1986 (cited on page 2).

[32] Lawrence C. Paulson. 'The foundation of a generic theorem prover'. In: *Journal of Automated Reasoning* 5.3 (Sept. 1, 1989), pp. 363–397. DOI: 10.1007/BF00248324 (cited on pages 2, 14).

[33] Patrick Cousot and Radhia Cousot. 'Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints'. In: *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*. Ed. by Robert M. Graham, Michael A. Harrison, and Ravi Sethi. ACM, 1977, pp. 238–252. DOI: 10.1145/512950.512973 (cited on page 2).

[34] N.G. Leveson and C.S. Turner. 'An investigation of the Therac-25 accidents'. In: *Computer* 26.7 (1993), pp. 18–41. DOI: 10.1109/MC.1993.274940 (cited on page 2).

[35] H. Gordon Rice. 'Classes of recursively enumerable sets and their decision problems'. In: *Transactions of the American Mathematical Society* 74 (1953), pp. 358–366 (cited on page 2).

[36] Patrick Cousot. *Principles of abstract interpretation*. MIT Press, 2021, pp. 1–819 (cited on page 2).

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 'Towards Deep Learning Models Resistant to Adversarial Attacks'. In: *International Conference on Learning Representations*. 2018 (cited on pages 2, 5, 15, 16, 85).

[38] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. 'AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation'. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 3–18 (cited on pages 3, 26).

[39] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. 'Fast and Effective Robustness Certification'. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 10802–10813 (cited on pages 3, 25–27, 35).

[40] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 'An Abstract Domain for Certifying Neural Networks'. In: *Proceedings of the ACM on Programming Languages* 3.POPL (2019), pp. 1–30. DOI: 10.1145/3290354 (cited on pages 3, 23, 25, 26, 69, 89).

[41] Claudio Ferrari, Mark Niklas Mueller, Nikola Jovanović, and Martin Vechev. 'Complete Verification via Multi-Neuron Relaxation Guided Branch-and-Bound'. In: *International Conference on Learning Representations* (2022) (cited on pages 3, 19, 20, 41, 69).

[42] Eric Wong and Zico Kolter. 'Provable defenses against adversarial examples via the convex outer adversarial polytope'. In: *International Conference on Machine Learning*. 2018 (cited on pages 3, 25–27).

[43] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. 'Towards fast computation of certified robustness for ReLU networks'. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5276–5285 (cited on pages 3, 25, 26).

[44] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. 'Scaling provable adversarial defenses'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018 (cited on page 3).

[45] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 'Efficient Neural Network Robustness Certification with General Activation Functions'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018 (cited on pages 4, 25–27, 69).

[46] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 'Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond'. In: *Neural Information Processing Systems*. 2020 (cited on pages 4, 18, 21, 26, 27, 50, 59, 61, 82, 97).

[47] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark W. Barrett, and Mykel J. Kochenderfer. 'Algorithms for Verifying Deep Neural Networks'. In: *Found. Trends Optim.* 4 (2019), pp. 244–404 (cited on page 4).

[48] Aws Albarghouthi. 'Introduction to Neural Network Verification'. In: *ArXiv* abs/2109.10317 (2021) (cited on page 4).

[49] Caterina Urban and Antoine Miné. 'A Review of Formal Methods applied to Machine Learning'. In: *CoRR* abs/2104.02466 (2021) (cited on pages 4, 97).

[50] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 'Evasion attacks against machine learning at test time'. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Springer. 2013 (cited on pages 4, 15).

[51] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 'Intriguing properties of neural networks'. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014 (cited on pages 4, 15).

[52] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. 'Perfectly Parallel Fairness Certification of Neural Networks'. In: *CoRR* abs/1912.02499 (2019) (cited on page 4).

[53] Joao Marques-Silva and Alexey Ignatiev. 'Delivering Trustworthy AI through Formal XAI'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.11 (June 2022), pp. 12342–12350. DOI: 10.1609/aaai.v36i11.21499 (cited on pages 4, 96).

[54] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 'Certified robustness to adversarial examples with differential privacy'. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 656–672 (cited on page 4).

[55] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. 'Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 241–249. DOI: 10.1109/CVPR42600.2020.00032 (cited on pages 4, 5, 39).

[56] Haoze Wu, Teruhiro Tagomori, Alexander Robey, Fengjun Yang, N. Matni, George Pappas, Hamed Hassani, Corina S. Păsăreanu, and Clark W. Barrett. 'Toward Certified Robustness Against Real-World Distribution Shifts'. In: *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)* (2022), pp. 537–553 (cited on pages 4, 5, 26, 27, 97).

[57] Denis Mazzucato and Caterina Urban. 'Reduced Products of Abstract Domains for Fairness Certification of Neural Networks'. In: *28th Static Analysis Symposium (SAS 2021)*. Chicago, United States, Oct. 2021 (cited on pages 4, 5, 27, 31, 33, 38).

[58] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 'Formal Security Analysis of Neural Networks Using Symbolic Intervals'. In: *Proceedings of the 27th USENIX Conference on Security Symposium*. SEC'18. Baltimore, MD, USA: USENIX Association, 2018, pp. 1599–1614 (cited on pages 5, 24, 31–33, 37, 42, 95).

[59] Augustin Lemesle, Julien Lehmann, and Tristan Le Gall. 'Neural Network Verification with PyRAT'. In: *ArXiv* abs/2410.23903 (2024) (cited on pages 5, 31, 35, 39, 89).

[60] Christopher Brix, Stanley Bak, Taylor T Johnson, and Haoze Wu. 'The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results'. In: *arXiv preprint arXiv:2412.19985* (2024) (cited on pages 5, 31, 32, 39, 95).

[61] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 'Adversarial training for free!' In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019 (cited on pages 5, 16, 46).

[62] Alessandro De Palma, Serge Durand, Zakaria Chihani, François Terrier, and Caterina Urban. 'On Using Certified Training towards Empirical Robustness'. In: *Transactions on Machine Learning Research* (2025) (cited on pages 6, 45).

[63] Caglar Aytekin. 'Neural networks are decision trees'. In: *arXiv preprint arXiv:2210.05189* (2022) (cited on page 6).

[64] Romain Xu-Darme. 'Algorithms and evaluation metrics for improving trust in machine learning : application to visual object recognition'. PhD thesis. Université Grenoble Alpes [2020-....], Nov. 2023 (cited on page 6).

[65] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. URL: https://arxiv.org/abs/1312.6034 (cited on pages 6, 80).

[66] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 'Axiomatic attribution for deep networks'. In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328 (cited on pages 6, 74, 80).

[67] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. *SmoothGrad: removing noise by adding noise*. 2017. URL: https://arxiv.org/abs/1706.03825 (cited on pages 6, 74, 80).

[68] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. 'Layer-wise relevance propagation for neural networks with local renormalization layers'. In: *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*. Springer. 2016, pp. 63–71 (cited on page 6).

[69] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. '" Why should i trust you?" Explaining the predictions of any classifier'. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cited on pages 6, 74).

[70] Xuanxiang Huang and Joao Marques-Silva. 'From Robustness to Explainability and Back Again'. In: *ArXiv* abs/2306.03048 (2023) (cited on pages 6, 74, 75).

[71] Min Wu, Haoze Wu, and Clark Barrett. 'VeriX: Towards Verified Explainability of Deep Neural Networks'. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cited on pages 6, 73–75, 80, 82, 89, 91, 96).

[72] Shahaf Bassan and Guy Katz. 'Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks'. In: *Tools and Algorithms for the Construction and Analysis of Systems: 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22–27, 2023, Proceedings, Part I*. Paris, France: Springer-Verlag, 2023, pp. 187–207. DOI: 10.1007/978-3-031-30823-9_10 (cited on pages 6, 73, 90, 96).

[73] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. 'A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks'. In: *International Conference on Learning Representations*. 2019 (cited on page 14).

[74] Guodong Zhang, James Martens, and Roger B Grosse. 'Fast Convergence of Natural Gradient Descent for Over-Parameterized Neural Networks'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019 (cited on page 14).

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 'Identity mappings in deep residual networks'. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer. 2016, pp. 630–645 (cited on pages 15, 47, 91).

[76] Sergey Ioffe and Christian Szegedy. 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift'. In: *International Conference on Machine Learning*. 2015 (cited on pages 15, 50).

[77] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958 (cited on page 15).

[78] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022 (cited on page 15).

[79] Francesco Croce and Matthias Hein. 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks'. In: *ICML*. 2020 (cited on pages 16, 69–71, 78, 86).

[80] Leslie Rice, Eric Wong, and Zico Kolter. 'Overfitting in adversarially robust deep learning'. In: *International conference on machine learning*. PMLR. 2020, pp. 8093–8104 (cited on pages 16, 47).

[81] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. 'Robust overfitting may be mitigated by properly learned smoothening'. In: *International Conference on Learning Representations*. 2020 (cited on page 16).

[82] Yifei Wang, Liangchen Li, Jiansheng Yang, Zhouchen Lin, and Yisen Wang. 'Balance, imbalance, and rebalance: Understanding robust overfitting from a minimax game perspective'. In: *Advances in neural information processing systems* 36 (2024) (cited on page 16).

[83] BS Vivek and R Venkatesh Babu. 'Single-step adversarial training with dropout scheduling'. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2020 (cited on page 16).

[84] Geon Yeong Park and Sang Wan Lee. 'Reliably fast adversarial training via latent adversarial perturbation'. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021 (cited on page 16).

[85] Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. 'Subspace adversarial training'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022 (cited on page 16).

[86] Theodoros Tsiligkaridis and Jay Roberts. 'Understanding and increasing efficiency of frank-wolfe adversarial training'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022 (cited on page 16).

[87] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and Venkatesh Babu R. 'Guided Adversarial Attack for Evaluating and Enhancing Adversarial Defenses'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 20297–20308 (cited on page 16).

[88] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and Venkatesh Babu Radhakrishnan. 'Towards Efficient and Effective Adversarial Training'. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021 (cited on page 16).

[89] Hoki Kim, Woojin Lee, and Jaewook Lee. 'Understanding catastrophic overfitting in single-step adversarial training'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 8119–8127 (cited on pages 16, 17).

[90] Zeinab Golgooni, Mehrdad Saberi, Masih Eskandar, and Mohammad Hossein Rohban. 'ZeroGrad: Costless conscious remedies for catastrophic overfitting in the FGSM adversarial training'. In: *Intelligent Systems with Applications* 19 (2023), p. 200258. DOI: https://doi.org/10.1016/j.iswa.2023.200258 (cited on pages 16, 17).

[91] Guillermo Ortiz-Jimenez, Pau de Jorge, Amartya Sanyal, Adel Bibi, Puneet K Dokania, Pascal Frossard, Grégory Rogez, and Philip Torr. 'Catastrophic overfitting can be induced with discriminative non-robust features'. In: *Transactions on Machine Learning Research* (2023) (cited on pages 17, 45, 46).

[92] Runqi Lin, Chaojian Yu, and Tongliang Liu. 'Eliminating catastrophic overfitting via abnormal adversarial examples regularization'. In: *Advances in Neural Information Processing Systems* (2023) (cited on pages 17, 53).

[93] Runqi Lin, Chaojian Yu, Bo Han, Hang Su, and Tongliang Liu. 'Layer-Aware Analysis of Catastrophic Overfitting: Revealing the Pseudo-Robust Shortcut Dependency'. In: *International Conference on Machine Learning* (2024) (cited on pages 17, 18, 53).

[94] Teruo Sunaga. 'Theory of an interval algebra and its application to numerical analysis'. In: *RAAG Memoirs* (1958) (cited on page 18).

[95] R.E. Moore. *Interval Analysis*. Prentice-Hall series in automatic computation. Prentice-Hall, 1966 (cited on page 18).

[96] A. Neumaier. 'The Wrapping Effect, Ellipsoid Arithmetic, Stability and Confidence Regions'. In: *Validation Numerics: Theory and Applications*. Ed. by R. Albrecht, G. Alefeld, and H. J. Stetter. Vienna: Springer Vienna, 1993, pp. 175–190. DOI: 10.1007/978-3-7091-6918-6_14 (cited on page 19).

[97] Jiameng Fan and Wenchao Li. 'Adversarial Training and Provable Robustness: A Tale of Two Objectives'. In: *AAAI Conference on Artificial Intelligence*. 2021 (cited on pages 19–21).

[98] Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 'Fast Certified Robust Training with Short Warmup'. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 18335–18349 (cited on pages 19, 21, 46, 51, 58, 59, 61, 64, 69, 70, 85, 86).

[99] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. 'On the effectiveness of interval bound propagation for training verifiably robust models'. In: *arXiv preprint arXiv:1810.12715* (2018) (cited on pages 19, 21, 69, 70, 73, 74).

[100] P Henriksen and A Lomuscio. 'DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis'. In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*. 2021 (cited on pages 19, 20, 40).

[101] Desheng Wang, Weidong Jin, Yunpu Wu, and Aamir Khan. 'Improving Global Adversarial Robustness Generalization With Adversarially Trained GAN'. In: *arXiv preprint arXiv:2103.04513* (2021) (cited on pages 19, 20).

[102] Mislav Balunovic and Martin Vechev. 'Adversarial Training and Provable Defenses: Bridging the Gap'. In: *International Conference on Learning Representations*. 2020 (cited on pages 19–21).

[103] Alessandro De Palma, Rudy Bunel, Krishnamurthy Dvijotham, M. Pawan Kumar, and Robert Stanforth. 'IBP Regularization for Verified Adversarial Robustness via Branch-and-Bound'. In: *ICML 2022 Workshop on Formal Verification of Machine Learning*. 2022 (cited on pages 19–21).

[104] Yuhao Mao, Mark Niklas Muller, Marc Fischer, and Martin T. Vechev. 'TAPS: Connecting Certified and Adversarial Training'. In: *ArXiv* abs/2305.04574 (2023) (cited on pages 19–22, 46, 58, 59, 85).

[105] Mark Niklas Müller, Franziska Eckert, Marc Fischer, and Martin Vechev. 'Certified Training: Small Boxes are All You Need'. In: *International Conference on Learning Representations*. 2023 (cited on pages 19–21, 46, 58, 59, 70, 85).

[106] Rich Caruana. 'Multitask Learning'. In: *Machine Learning* 28.1 (July 1, 1997), pp. 41–75. DOI: 10.1023/A:1007379606734 (cited on page 20).

[107] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. 'Towards Stable and Efficient Training of Verifiably Robust Neural Networks'. In: *International Conference on Learning Representations*. 2020 (cited on pages 20, 21, 25, 70, 86, 90, 91).

[108] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. 'Rethinking Lipschitz Neural Networks for Certified L-infinity Robustness'. In: *arXiv preprint arXiv:2210.01787* (2022) (cited on page 21).

[109] Yuhao Mao, Mark Niklas Müller, Marc Fischer, and Martin Vechev. 'Understanding certified training with interval bound propagation'. In: *International Conference on Learning Representations* (2024) (cited on page 21).

[110] Jianlin Li, Jiangchao Liu, Pengfei Yang, Liqian Chen, Xiaowei Huang, and Lijun Zhang. 'Analyzing Deep Neural Networks with Symbolic Propagation: Towards Higher Precision and Faster Verification'. In: *Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings*. Ed. by Bor-Yuh Evan Chang. Vol. 11822. Lecture Notes in Computer Science. Springer, 2019, pp. 296–319. DOI: 10.1007/978-3-030-32304-2_15 (cited on page 24).

[111] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 'Efficient Formal Safety Analysis of Neural Networks'. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 6367–6377 (cited on pages 25, 40).

[112] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 'Boosting Robustness Certification of Neural Networks'. In: *International Conference on Learning Representations (ICLR)*. 2019 (cited on pages 25, 32, 35, 40).

[113] Antoine Girard. 'Reachability of Uncertain Linear Systems Using Zonotopes'. In: *Hybrid Systems: Computation and Control*. Ed. by Manfred Morari and Lothar Thiele. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 291–305 (cited on page 26).

[114] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. 'The Zonotope Abstract Domain Taylor1+'. In: *Computer Aided Verification*. Ed. by Ahmed Bouajjani and Oded Maler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 627–633 (cited on page 26).

[115] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. 'A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 9835–9846 (cited on pages 26, 41).

[116] L. Li, Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. 'SoK: Certified Robustness for Deep Neural Networks'. In: *2023 IEEE Symposium on Security and Privacy (SP)* (2020), pp. 1289–1310 (cited on page 26).

[117] Andrew L. Maas. 'Rectifier Nonlinearities Improve Neural Network Acoustic Models'. In: 2013 (cited on page 26).

[118] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification'. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034 (cited on pages 26, 27).

[119] Alex Krizhevsky. 'Convolutional deep belief networks on cifar-10'. In: *Unpublished* (2010) (cited on pages 26, 27).

[120] P Henriksen and A Lomuscio. 'Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search'. In: *ECAI 2020 proceedings*. 2020 (cited on pages 26, 27, 40).

[121] Zhaodi Zhang, Yiting Wu, Siwen Liu, Jing Liu, and Min Zhang. 'Provably Tightest Linear Approximation for Robustness Verification of Sigmoid-like Neural Networks'. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (2022) (cited on pages 26, 27).

[122] Yunruo Zhang, Lujia Shen, Shanqing Guo, and Shouling Ji. 'GaLileo: General Linear Relaxation Framework for Tightening Robustness Certification of Transformers'. In: *AAAI Conference on Artificial Intelligence*. 2024 (cited on pages 26, 27).

[123] Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. 'Neural Network Verification with Branch-and-Bound for General Nonlinearities'. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Arie Gurfinkel and Marijn Heule. Cham: Springer Nature Switzerland, 2025, pp. 315–335 (cited on pages 26, 27, 41).

[124] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Neural Information Processing Systems* (2019) (cited on pages 26, 27, 50).

[125] Nikola Jovanović, Mislav Balunovic, Maximilian Baader, and Martin Vechev. 'On the Paradox of Certified Training'. In: *Transactions on Machine Learning Research* (2022) (cited on pages 27, 70).

[126] Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz. 'An Abstraction-Based Framework for Neural Network Verification'. In: *CoRR* abs/1910.14574 (2019) (cited on pages 27, 41).

[127] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. 'Open-and closed-loop neural network verification using polynomial zonotopes'. In: *NASA Formal Methods Symposium*. Springer. 2023, pp. 16–36 (cited on page 27).

[128] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. 'Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers'. In: *International Conference on Learning Representations*. 2021 (cited on page 28).

[129] Serge Durand, Augustin Lemesle, Zakaria Chihani, Caterina Urban, and François Terrier. 'ReCIPH: Relational Coefficients for Input Partitioning Heuristic'. In: *workshop on Formal Verification of Machine Learning (WFVML 2022)* (2022) (cited on page 31).

[130] Kyle D. Julian, Jessica Lopez, Jeffrey S. Brush, Michael P. Owen, and Mykel J. Kochenderfer. 'Policy compression for aircraft collision avoidance systems'. In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. 2016, pp. 1–10. DOI: 10.1109/DASC.2016.7778091 (cited on pages 31, 35).

[131] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'. In: *arXiv preprint arXiv:1702.01135* (2017) (cited on pages 31, 32, 35, 40, 41).

[132] Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. 'Evaluating the Safety of Deep Reinforcement Learning Models using Semi-Formal Verification'. In: *CoRR* abs/2010.09387 (2020) (cited on page 33).

[133] Kyle D. Julian, Mykel J. Kochenderfer, and Michael P. Owen. 'Deep neural network compression for aircraft collision avoidance systems'. In: *Journal of Guidance Control and Dynamics* 42.3 (2019), pp. 598–608. DOI: 10.2514/1.G003724 (cited on page 35).

[134] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 'Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks'. In: *Computer Aided Verification*. Ed. by Rupak Majumdar and Viktor Kunčak. Cham: Springer International Publishing, 2017, pp. 97–117 (cited on pages 35, 37, 76, 95).

[135] Djoni E. Sidarta, Jim O'Sullivan, and Ho-Joon Lim. 'Damage Detection of Offshore Platform Mooring Line Using Artificial Neural Network'. In: vol. Volume 1: Offshore Technology. International Conference on Offshore Mechanics and Arctic Engineering. June 2018. DOI: 10.1115/OMAE2018-77084 (cited on page 37).

[136] Djoni E. Sidarta, Ho-Joon Lim, Johyun Kyoung, Nicolas Tcherniguin, Timothee Lefebvre, and Jim O'Sullivan. 'Detection of Mooring Line Failure of a Spread-Moored FPSO: Part 1 — Development of an Artificial Neural Network Based Model'. In: vol. Volume 1: Offshore Technology; Offshore Geotechnics. International Conference on Offshore Mechanics and Arctic Engineering. V001T01A042. June 2019. DOI: 10.1115/OMAE2019-96288 (cited on page 37).

[137] Luca Pulina and Armando Tacchella. 'Challenging SMT solvers to verify neural networks'. In: *AI Commun.* 25.2 (2012), pp. 117–135. DOI: 10.3233/AIC-2012-0525 (cited on page 40).

[138] Rüdiger Ehlers. 'Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks'. In: *ArXiv* abs/1705.01320 (2017) (cited on page 40).

[139] Xiaowei Huang, M. Kwiatkowska, Sen Wang, and Min Wu. 'Safety Verification of Deep Neural Networks'. In: *International Conference on Computer Aided Verification*. 2016 (cited on page 40).

[140] Vincent Tjeng, Kai Xiao, and Russ Tedrake. 'Evaluating Robustness of Neural Networks with Mixed Integer Programming'. In: International Conference on Learning Representations (ICLR). 2019. (Visited on 06/19/2019) (cited on page 40).

[141] Philipp Kern, Marko Kleine Büning, and Carsten Sinz. 'Optimized symbolic interval propagation for neural network verification'. In: *arXiv preprint arXiv:2212.08567* (2022) (cited on page 40).

[142] Lukas Koller, Tobias Ladner, and Matthias Althoff. 'Out of the Shadows: Exploring a Latent Space for Neural Network Verification'. In: *ArXiv* abs/2505.17854 (2025) (cited on page 40).

[143] Matthias Althoff. 'An Introduction to CORA 2015'. In: *EPiC Series in Computing*. EasyChair, 2015. DOI: 10.29007/zbkv (cited on page 40).

[144] Vicenç Rúbies Royo, Roberto Calandra, Dusan M. Stipanovic, and Claire Tomlin. 'Fast Neural Network Verification via Shadow Prices'. In: *CoRR* abs/1902.07247 (2019) (cited on page 40).

[145] Stanley Bak, Hoang-Dung Tran, Kerianne Hobbs, and Taylor T. Johnson. 'Improved Geometric Path Enumeration for Verifying ReLU Neural Networks'. In: *32nd International Conference on Computer-Aided Verification (CAV)*. Sept. 2020. DOI: 10.1007/978-3-030-53288-8_4 (cited on pages 40, 41).

[146] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 'Star-Based Reachability Analysis of Deep Neural Networks'. In: *Formal Methods – The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings*. Porto, Portugal: Springer-Verlag, 2019, pp. 670–686. DOI: 10.1007/978-3-030-30942-8_39 (cited on page 41).

[147] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel Kochenderfer, and Clark Barrett. 'The Marabou Framework for Verification and Analysis of Deep Neural Networks'. In: *To appear in Proceedings of the* 31^{st} *International Conference on Computer Aided Verification (CAV '19)*. Sept. 2019 (cited on pages 41, 75).

[148] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H.S. Torr, Pushmeet Kohli, and M. Pawan Kumar. 'Branch and Bound for Piecewise Linear Neural Network Verification'. In: *Journal of Machine Learning Research* 21.42 (2020), pp. 1–39 (cited on page 41).

[149] Jingyue Lu and M. Pawan Kumar. 'Neural Network Branching for Neural Network Verification'. In: *International Conference on Learning Representations*. 2020 (cited on pages 41, 96).

[150] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. 'A dual approach to scalable verification of deep networks'. In: *Conference on Uncertainty in Artificial Intelligence* (2018) (cited on page 41).

[151] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. 'Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification'. In: *Neural Information Processing Systems* (2021) (cited on page 41).

[152] Alessandro De Palma, Rudy Bunel, Alban Desmaison, Krishnamurthy Dvijotham, Pushmeet Kohli, Philip H. S. Torr, and M. Pawan Kumar. *Improved Branch and Bound for Neural Network Verification via Lagrangian Decomposition*. 2021. URL: https://arxiv.org/abs/2104.06718 (cited on page 41).

[153] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. 'A Branch and Bound Framework for Stronger Adversarial Attacks of ReLU Networks'. In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, Sept. 2022, pp. 26591–26604 (cited on page 41).

[154] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. 'General Cutting Planes for Bound-Propagation-Based Neural Network Verification'. In: *Neural Information Processing Systems* (2022) (cited on page 41).

[155] Duo Zhou, Christopher Brix, Grani A Hanasusanto, and Huan Zhang. 'Scalable Neural Network Verification with Branch-and-bound Inferred Cutting Planes'. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang. Vol. 37. Curran Associates, Inc., 2024, pp. 29324–29353 (cited on page 41).

[156] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. 'Beyond the single neuron convex barrier for neural network certification'. In: *Neural Information Processing Systems* (2019) (cited on page 41).

[157] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. 'Strong mixed-integer programming formulations for trained neural networks'. In: *Mathematical Programming* (2020) (cited on page 41).

[158] Christian Tjandraatmadja, Ross Anderson, Joey Huchette, Will Ma, Krunal Patel, and Juan Pablo Vielma. 'The Convex Relaxation Barrier, Revisited: Tightened Single-Neuron Relaxations for Neural Network Verification'. In: *Neural Information Processing Systems* (2020) (cited on page 41).

[159] Alessandro De Palma, Harkirat Singh Behl, Rudy Bunel, Philip H. S. Torr, and M. Pawan Kumar. 'Scaling the Convex Barrier with Sparse Dual Algorithms'. In: *Journal of Machine Learning Research* (2024) (cited on pages 41, 70).

[160] Alessandro De Palma, Harkirat Singh Behl, Rudy Bunel, Philip H. S. Torr, and M. Pawan Kumar. 'Scaling the Convex Barrier with Active Sets'. In: *International Conference on Learning Representations* (2021) (cited on page 41).

[161] Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy R Bunel, Shreya Shankar, Jacob Steinhardt, Ian Goodfellow, Percy S Liang, and Pushmeet Kohli. 'Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 5318–5331 (cited on page 42).

[162] Hong-Ming Chiu, Hao Chen, Huan Zhang, and Richard Y. Zhang. 'SDP-CROWN: Efficient Bound Propagation for Neural Network Verification with Tightness of Semidefinite Programming'. In: *Forty-second International Conference on Machine Learning*. 2025 (cited on page 42).

[163] Zhouxing Shi, Yihan Wang, Huan Zhang, Zico Kolter, and Cho-Jui Hsieh. 'Efficiently computing local lipschitz constants of neural networks via bound propagation'. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2022 (cited on page 42).

[164] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. 'Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019 (cited on page 42).

[165] Yuhao Zhang, Aws Albarghouthi, and Loris D'antoni. 'Certified Robustness to Programmable Transformations in LSTMs'. In: *ArXiv* abs/2102.07818 (2021) (cited on page 42).

[166] Bohang Zhang, Du Jiang, Di He, and Liwei Wang. 'Rethinking Lipschitz Neural Networks and Certified Robustness: A Boolean Function Perspective'. In: *Neural Information Processing Systems*. 2022 (cited on page 42).

[167] Cem Anil, James Lucas, and Roger Grosse. 'Sorting out Lipschitz function approximation'. In: *International conference on machine learning*. PMLR. 2019, pp. 291–301 (cited on page 42).

[168] Alexandre Araujo, Aaron J Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. 'A Unified Algebraic Perspective on Lipschitz Neural Networks'. In: *The Eleventh International Conference on Learning Representations*. 2023 (cited on page 42).

[169] Klas Leino, Zifan Wang, and Matt Fredrikson. 'Globally-robust neural networks'. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6212–6222 (cited on page 42).

[170] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 'Reading Digits in Natural Images with Unsupervised Feature Learning'. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011 (cited on pages 47, 50).

[171] A. Krizhevsky and G. Hinton. 'Learning multiple layers of features from tiny images'. In: *Master's thesis, Department of Computer Science, University of Toronto* (2009) (cited on page 50).

[172] Akhilan Boopathy, Lily Weng, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Luca Daniel. 'Fast Training of Provably Robust Neural Networks by SingleProp'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 6803–6811 (cited on page 69).

[173] Piersilvio De Bartolomeis, Jacob Clarysse, Amartya Sanyal, and Fanny Yang. 'How robust accuracy suffers from certified training with convex relaxations'. In: *arXiv preprint arXiv:2306.06995* (2023) (cited on pages 69, 70).

[174] Yuhao Mao, Stefan Balauca, and Martin Vechev. 'CTBench: A Library and Benchmark for Certified Training'. In: *Forty-second International Conference on Machine Learning*. 2025 (cited on pages 69–71, 76, 88).

[175] Joao Marques-Silva. 'Logic-Based Explainability in Machine Learning'. In: *ArXiv* abs/2211.00541 (2022) (cited on pages 73–75).

[176] Min Wu, Xiaofu Li, Haoze Wu, and Clark W. Barrett. 'Better Verified Explanations with Applications to Incorrectness and Out-of-Distribution Detection'. In: *ArXiv* abs/2409.03060 (2024) (cited on pages 73, 76, 77, 81, 82, 89, 90).

[177] Dorin Doncenco, Julien Girard-Satabin, Romain Xu-Darme, and Zakaria Chihani. 'A Dive into Formal Explainable Attributions for Image Classification'. In: *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*. Bologna, Italy: IOS Press, 2025 (cited on pages 73, 81, 89, 90).

[178] Yacine Izza, Xuanxiang Huang, Antonio Morgado, Jordi Planes, Alexey Ignatiev, and Joao Marques-Silva. 'Distance-restricted explanations: theoretical underpinnings & efficient implementation'. In: *arXiv preprint arXiv:2405.08297* (2024) (cited on pages 73, 90).

[179] Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 'Fast certified robust training with short warmup'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18335–18349 (cited on pages 73, 74).

[180] Matthew Mirman, Gagandeep Singh, and Martin Vechev. 'A Provable Defense for Deep Residual Networks'. In: 2019 (cited on pages 73, 74).

[181] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T Johnson. 'The Third International Verification of Neural Networks Competition (VNN-COMP 2022): Summary and Results'. In: *arXiv preprint arXiv:2212.10376* (2022) (cited on pages 73, 74).

[182] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 3rd ed. 2025 (cited on pages 73, 74).

[183] Melkamu Mersha, Khang Lam, Joseph Wood, Ali K. AlShami, and Jugal Kalita. 'Explainable artificial intelligence: A survey of needs, techniques, applications, and future direction'. In: *Neurocomputing* 599 (Sept. 2024), p. 128111. DOI: 10.1016/j.neucom.2024.128111 (cited on pages 73, 74).

[184] Scott M Lundberg and Su-In Lee. 'A unified approach to interpreting model predictions'. In: *Advances in neural information processing systems* 30 (2017) (cited on page 74).

[185] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 'Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74 (cited on page 74).

[186] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 'On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation'. In: *PLoS ONE* 10 (2015) (cited on page 74).

[187] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. *Striving for Simplicity: The All Convolutional Net*. 2015. URL: https://arxiv.org/abs/1412.6806 (cited on page 74).

[188] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 'Learning important features through propagating activation differences'. In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153 (cited on page 74).

[189] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 'Towards better understanding of gradient-based attribution methods for Deep Neural Networks'. In: *International Conference on Learning Representations*. 2018 (cited on page 74).

[190] Nina Narodytska, Aditya Shrotri, Kuldeep S. Meel, Alexey Ignatiev, and Joao Marques-Silva. 'Assessing Heuristic Machine Learning Explanations with Model Counting'. In: *Theory and Applications of Satisfiability Testing – SAT 2019*. Ed. by Mikoláš Janota and Inês Lynce. Cham: Springer International Publishing, 2019, pp. 267–278 (cited on page 74).

[191] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 'Sanity checks for saliency maps'. In: *Advances in neural information processing systems* 31 (2018) (cited on page 74).

[192] Romain Xu-Darme, Jenny Benois-Pineau, Romain Giot, Georges Quénot, Zakaria Chihani, Marie-Christine Rousset, and Alexey Zhukov. 'On the stability, correctness and plausibility of visual explanation methods based on feature importance'. In: *Proceedings of the 20th International Conference on Content-based Multimedia Indexing*. 2023, pp. 119–125 (cited on page 74).

[193] Anna Hedström, Philine Lou Bommer, Kristoffer Knutsen Wickstrøm, Wojciech Samek, Sebastian Lapuschkin, and Marina MC Höhne. 'The Meta-Evaluation Problem in Explainable AI: Identifying Reliable Estimators with MetaQuantus'. In: *Transactions on Machine Learning Research* (2023) (cited on page 74).

[194] Andy Shih, Arthur Choi, and Adnan Darwiche. 'A symbolic approach to explaining bayesian network classifiers'. In: *arXiv preprint arXiv:1805.03364* (2018) (cited on pages 74, 75).

[195] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. 'Abduction-based explanations for machine learning models'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 1511–1519 (cited on pages 74, 75).

[196] Emanuele La Malfa, Rhiannon Michelmore, Agnieszka M. Zbrzezny, Nicola Paoletti, and Marta Kwiatkowska. 'On Guaranteed Optimal Robust Explanations for NLP Models'. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 2658–2665. DOI: 10.24963/ijcai.2021/366 (cited on pages 74, 75, 96).

[197] Shahaf Bassan, Guy Amir, and Guy Katz. 'Local vs. Global Interpretability: A Computational Complexity Perspective'. In: *Forty-first International Conference on Machine Learning*. 2024 (cited on page 75).

[198] Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark Barrett. 'Marabou 2.0: A Versatile Formal Analyzer of Neural Networks'. In: *Computer Aided Verification: 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part II*. Montreal, QC, Canada: Springer-Verlag, 2024, pp. 249–264. DOI: 10.1007/978-3-031-65630-9_13 (cited on page 76).

[199] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014) (cited on page 86).

[200] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 'The German Traffic Sign Recognition Benchmark: A multi-class classification competition'. In: *The 2011 International Joint Conference on Neural Networks*. 2011, pp. 1453–1460. DOI: 10.1109/IJCNN.2011.6033395 (cited on page 89).

[201] Ulrich Junker. 'QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems'. In: *Proceedings of the 19th National Conference on Artifical Intelligence*. AAAI'04. San Jose, California: AAAI Press, 2004, pp. 167–172 (cited on page 90).

[202] Shahaf Bassan, Yizhak Yisrael Elboher, Tobias Ladner, Matthias Althoff, and Guy Katz. 'Explaining, Fast and Slow: Abstraction and Refinement of Provable Explanations'. In: *Forty-second International Conference on Machine Learning*. 2025 (cited on page 90).

[203] Tobias Ladner and Matthias Althoff. 'Fully Automatic Neural Network Reduction for Formal Verification'. In: *Transactions on Machine Learning Research* (2025) (cited on page 90).

[204] Shahaf Bassan, Ron Eliav, and Shlomit Gur. 'Explain Yourself, Briefly! Self-Explaining Neural Networks with Concise Sufficient Reasons'. In: *The Thirteenth International Conference on Learning Representations*. 2025 (cited on page 90).

[205] Ruth C. Fong and Andrea Vedaldi. 'Interpretable Explanations of Black Boxes by Meaningful Perturbation'. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.371 (cited on page 91).

[206] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 'Anchors: high-precision model-agnostic explanations'. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018 (cited on page 91).

[207] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. 'What made you do this? Understanding black-box decisions with sufficient input subsets'. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, Apr. 2019, pp. 567–576 (cited on page 91).

[208] Thomas Fel, Mélanie Ducoffe, David Vigouroux, Rémi Cadène, Mikael Capelle, Claire Nicodème, and Thomas Serre. 'Don't Lie to Me! Robust and Efficient Explainability with Verified Perturbation Analysis'. In: *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*. Vancouver, Canada: IEEE, June 2023. DOI: 10.1109/cvpr52729.2023.01550 (cited on page 91).

[209] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017) (cited on page 97).

[210] Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin T. Vechev. 'Fast and precise certification of transformers'. In: *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (2021) (cited on page 97).

[211] Satoshi Munakata, Caterina Urban, Haruki Yokoyama, Koji Yamamoto, and Kazuki Munakata. 'Verifying Attention Robustness of Deep Neural Networks against Semantic Perturbations'. In: *ArXiv* abs/2207.05902 (2022) (cited on page 97).

[212] Daqian Shao, Lukas Fesser, and Marta Z. Kwiatkowska. 'STR-Cert: Robustness Certification for Deep Text Recognition on Deep Learning Pipelines and Vision Transformers'. In: *ArXiv* abs/2401.05338 (2023) (cited on page 97).

[213] Anagha Athavale, Ezio Bartocci, Maria Christakis, Matteo Maffei, Dejan Nickovic, and Georg Weissenbacher. 'Verifying Global Two-Safety Properties in Neural Networks with Confidence'. In: *Computer Aided Verification*. Ed. by Arie Gurfinkel and Vijay Ganesh. Cham: Springer Nature Switzerland, 2024, pp. 329–351 (cited on page 97).

[214] David Boetius and Stefan Leue. *Verifying Global Neural Network Specifications using Hyperproperties*. 2023. URL: https://arxiv.org/abs/2306.12495 (cited on page 97).

[215] Debangshu Banerjee, Changming Xu, and Gagandeep Singh. 'Input-Relational Verification of Deep Neural Networks'. In: *Proc. ACM Program. Lang.* 8.PLDI (June 2024). DOI: 10.1145/3656377 (cited on page 97).

[216] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 'Adversarial examples are not bugs, they are features'. In: *Advances in neural information processing systems* 32 (2019) (cited on page 97).

[217] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola Schönlieb. 'On the Connection Between Adversarial Robustness and Saliency Map Interpretability'. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1823–1832 (cited on page 97).

[218] Prasad Chalasani, Jiefeng Chen, Amrita Roy Chowdhury, Xi Wu, and Somesh Jha. 'Concise Explanations of Neural Networks using Adversarial Training'. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 1383–1391 (cited on page 97).

[219] Suraj Srinivas, Sebastian Bordt, and Himabindu Lakkaraju. 'Which Models have Perceptually-Aligned Gradients? An Explanation via Off-Manifold Robustness'. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cited on pages 97, 98).

[220] Andrew Slavin Ross and Finale Doshi-Velez. 'Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients'. In: *AAAI Conference on Artificial Intelligence*. 2017 (cited on page 97).

[221] Akhilan Boopathy, Sijia Liu, Gaoyuan Zhang, Cynthia Liu, Pin-Yu Chen, Shiyu Chang, and Luca Daniel. 'Proper Network Interpretability Helps Adversarial Robustness in Classification'. In: *International Conference on Machine Learning*. 2020 (cited on page 97).

[222] Roy Ganz, Bahjat Kawar, and Michael Elad. 'Do Perceptually Aligned Gradients Imply Robustness?' In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 10628–10648 (cited on page 97).

[223] Sunghwan Joo, SeokHyeon Jeong, Juyeon Heo, Adrian Weller, and Taesup Moon. 'Towards More Robust Interpretation via Local Gradient Alignment'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.7 (June 2023), pp. 8168–8176. DOI: 10.1609/aaai.v37i7.25986 (cited on page 97).

[224] Debangshu Banerjee, Avaljot Singh, and Gagandeep Singh. 'Interpreting Robustness Proofs of Deep Neural Networks'. In: *The Twelfth International Conference on Learning Representations*. 2024 (cited on page 97).

[225] Mathieu Serrurier, Franck Mamalet, Thomas Fel, Louis Béthune, and Thibaut Boissin. 'On the explainable properties of 1-Lipschitz Neural Networks: An Optimal Transport Perspective'. In: 2022 (cited on page 97).