Probabilistically Tightened Linear Relaxation-based Perturbation Analysis for Neural Network Verification

LUCA MARZARI, Department of Computer Science, University of Verona, Italy
FERDINANDO CICALESE, Department of Computer Science, University of Verona, Italy
ALESSANDRO FARINELLI, Department of Computer Science, University of Verona, Italy

We present **P**robabilistically **T**ightened **Li**near **R**elaxation-based **P**erturbation **A**nalysis (PT-LiRPA), a novel framework that combines over-approximation techniques from LiRPA-based approaches with a sampling-based method to compute tight intermediate reachable sets. In detail, we show that with negligible computational overhead, PT-LiRPA exploiting the estimated reachable sets, significantly tightens the lower and upper linear bounds of a neural network's output, reducing the computational cost of formal verification tools while providing probabilistic guarantees on verification soundness. Extensive experiments on standard formal verification benchmarks, including the International Verification of Neural Networks Competition, show that our PT-LiRPA-based verifier improves robustness certificates, i.e., the certified lower bound of ϵ perturbation tolerated by the models, by up to 3.31X and 2.26X compared to related work. Importantly, our probabilistic approach results in a valuable solution for challenging competition entries where state-of-the-art formal verification methods fail, allowing us to provide answers with high confidence (i.e., at least 99%).

JAIR Track: Integration of Logical Constraints in Deep Learning

JAIR Associate Editor: Matteo Zavatteri

JAIR Reference Format:
, Luca Marzari, Ferdinando Cicalese, and Alessandro Farinelli. 2025. Probabilistically Tightened Linear Relaxation-based Perturbation Analysis for Neural Network Verification. *Journal of Artificial Intelligence Research* 84, Article 30 (December 2025), 34 pages. DOI: 10.1613/jair.1.20808

1 Introduction

Deep neural networks (DNNs) and recently large language models (LLMs) (Vaswani et al. 2017) have revolutionized various fields, from healthcare and finance to natural language processing, enabling remarkable capabilities, for instance, in image recognition (O'Shea and Nash 2015), robotic tasks such as manipulation (Marzari, Pore, et al. 2021; Rajeswaran et al. 2018) and navigation (Marzari, Cicalese, et al. 2025; Marzari, Corsi, Marchesini, and Farinelli 2022; Marzari, Donti, et al. 2025; Tai et al. 2017). Nonetheless, their opacity and vulnerability to the so-called "adversarial inputs" (Amir et al. 2023; Szegedy et al. 2013) have also raised significant concerns, especially when deployed in safety-critical applications such as autonomous driving or medical diagnosis. Consequently, developing methods to certify the safety aspect of these models is crucial. Achieving provable safety guarantees involves employing formal verification (FV) of neural network techniques (Liu et al. 2021), which mathematically ensures that a system will never produce undesired outcomes in all configurations tested.
In this work, we focus on robustness verification, which aims to guarantee that a model's output remains

Authors' Contact Information:, ORCID: 0000-0002-0069-0182; Luca Marzari, luca.marzari@univr.it, Department of Computer Science, University of Verona, Verona, Italy, ORCID: 0000-0003-1652-0599; Ferdinando Cicalese, ferdinando.cicalese@univr.it, Department of Computer Science, University of Verona, Verona, Italy, ORCID: 0000-0002-2592-5814; Alessandro Farinelli, alessandro.farinelli@univr.it, Department of Computer Science, University of Verona, Verona, Italy.

DOI: 10.1613/jair.1.20808

consistently robust under small predefined input perturbations around a given point $\boldsymbol{x}_0$, typically defined as an ℓ_p ball of ε radius around the input $\boldsymbol{x}_0$, i.e., the set $C = C_{\boldsymbol{x}_0,\varepsilon} = \{\boldsymbol{x} | \ ||\boldsymbol{x} - \boldsymbol{x}_0||_p \leq \varepsilon\}$. A standard approach to proving the robustness property is to first encode the desired output of the classifier as the sign of the value f computed by a single (output) node, i.e., so that the correct classification in $\boldsymbol{x}_0$ corresponds to $f(\boldsymbol{x}_0) > 0$. Then, the robustness verification becomes equivalent to checking that $\min_{\boldsymbol{x} \in C} f(\boldsymbol{x})$, is positive (S. Wang, H. Zhang, et al. 2021). However, due to the non-linear and non-convex nature of the DNN, solving this problem is, in general, NP-hard (Katz et al. 2017; L. Weng, H. Zhang, et al. 2018). A recent line of works called linear relaxation-based perturbation analysis (LiRPA) algorithms (S. Wang, H. Zhang, et al. 2021; Xu, Shi, et al. 2020; Xu, H. Zhang, et al. 2021; H. Zhang, T.-W. Weng, et al. 2018) proposes a formal analysis based on a sound linear relaxation of the DNN. The idea is to compute relaxations of all *sources of non-linearity* in the network so as to obtain two linear functions providing respectively a lower and an upper bound of the DNN's output, which are then used in the robustness certification. Nonetheless, these methods use overapproximation techniques to satisfy the worst-case setting, where no exceptions are allowed. While this strict approach ensures absolute safety within the certified input space, it faces crucial limitations. In fact, the efficiency of LiRPA approaches, and, in general, any FV methods, scales poorly with the size of the model. Additionally, formal methods fail to offer meaningful robustness information when input perturbations exceed the certified bounds. As discussed in (L. Weng, Chen, et al. 2019), relying solely on formal verification can lead to overly conservative outcomes, particularly when adversarial examples are rare or when prior knowledge of the input distribution is available. This stems from the exact nature of formal solvers, which treat all violations equally: even a single adversarial input—such as a one-pixel change—invalidates the entire region, without distinguishing between isolated, low-probability cases and large, semantically significant unsafe regions.

Similar to recent approaches (Cohen et al. 2019; L. Weng, Chen, et al. 2019), to overcome these critical limitations, we propose a probabilistic perspective that allows for an infinitesimal trade-off in certainty while providing the likelihood of such violations, enabling more nuanced and practical assessments of robustness that better reflect real-world risks. Importantly, the goal of this work is not to compare probabilistic methods with provable ones, as they are fundamentally different in nature. Rather, our objective is to propose a novel, complementary solution that enhances formal methods by offering additional safety information about the models under evaluation for particular challenging instances to be verified.

Building on this foundation, we explore Xu, H. Zhang, et al. (2021) speculation that estimates as tightly as possible reachable sets, i.e., the range of possible values at each hidden node given bounded inputs, could significantly enhance verification efficiency by yielding sharper final linear bounds. In this perspective, we address two key research questions: (i) *How can we compute (probabilistic) reachable sets that yield tighter intermediate and output bounds than existing methods, while remaining computationally efficient?* (ii) *What theoretical guarantees can we establish for linearized layers using (probabilistically sound) reachable set?*

Our Contributions. We propose **P**robabilistically **T**ightened **Li**near **R**elaxation-based **P**erturbation **A**nalysis (`PT-LiRPA`), a novel framework for computing tight linear lower and upper bounds combining existing LiRPA methods with a sampling-based reachable set estimation strategy. Unlike other related probabilistic works (Cohen et al. 2019; L. Weng, Chen, et al. 2019), our approach does not rely on specific input distributions or attacks. Specifically, in this work, we start considering statistical results on tolerance limits (Wilks 1942) to provide probabilistic guarantees on estimating the minimum and maximum values of a neural network's output. Using the approach of Wilks (1942), we are guaranteed that, for any $R, \psi \in (0, 1)$, with probability ψ, at most, a fraction $(1 - R)$ of points from a possibly infinitely large sample in the perturbation region C may violate the estimated bounds obtained from an initial sample, whose size n is explicitly computable from the desired parameters ψ and R. Hence, our intuition is to extend this approach to also compute an estimation of reachable sets with a specified confidence level. Nonetheless, while the result of (Wilks 1942) quantitatively bounds the error of the

sample-based procedure by predicting the fraction of potential violations in future samples, it results in "weaker" guarantees with respect to other known probabilistic approaches. In particular, this approach does not provide information on the magnitude of these violations with respect to the estimated bounds. Specifically, in any estimated reachable set for a possibly existing fraction $(1-R)$ of points drawn from C, the probability of violating the bound might in principle be uncontrollably large. To address such an issue, following results on *extreme value theory* (EVT) (De Haan 1981; Haan and Ferreira 2006), we extend Wilks (1942)' probabilistic guarantees and present two novel bounds on the magnitude of potential errors between the true minimum (Theorem 3.5) and the sample-based estimated one (Theorem 3.7). Notably, our final result proves that with negligible computational overhead, each node's reachable set computed using random samples represents with high probability the actual domain of that node for any $x \in C$. Hence, we show that by employing this sampling-based procedure to compute probabilistically tight reachable set bounds in the neural network and integrating these into the linearization used by LiRPA-based formal verification methods, we are able to obtain significantly tighter lower and upper linear bounds of a neural network's output, while preserving the verification soundness for any input in the perturbation region and specified confidence level.

To assess the benefit and effectiveness of our novel framework, we perform an extensive empirical evaluation. We first compare our approach on neural networks trained on MNIST and CIFAR datasets with PROVEN (L. Weng, Chen, et al. 2019) and Randomized Smoothing (Cohen et al. 2019), the most closely related probabilistic verification approaches. In addition, as a ground truth, we also consider a set of state-of-the-art worst-case robustness verification approaches, namely CROWN(H. Zhang, T.-W. Weng, et al. 2018), α-CROWN (Xu, H. Zhang, et al. 2021), β-CROWN (S. Wang, H. Zhang, et al. 2021), and GCP-CROWN (H. Zhang, S. Wang, et al. 2022). This first set of experiments shows that with a very high confidence (i.e., $\geq 99\%$), `PT-LiRPA` improves the certified lower bound of ε perturbation tolerated by the models up to 3.31x and 2.26x compared with both the corresponding probabilistic approaches of (Cohen et al. 2019; L. Weng, Chen, et al. 2019) and up to 3.62x w.r.t. the worst-case analysis results. Finally, in the second set of experiments, we demonstrate that for challenging instances from the International Verification of Neural Networks Competition (VNN-COMP) (Brix et al. 2023; Müller et al. 2022), where state-of-the-art formal verification methods fail to produce a conclusive result, our `PT-LiRPA`, with a quantifiable level of confidence represents a valuable resource in providing safety information. The paper is structured as follows. Section 2 introduces the necessary background on robustness verification and explains how to perform linear relaxation-based perturbation analysis in a Rectified Linear Unit (ReLU)-based deep neural network. We also include a detailed discussion comparing our probabilistic guarantees with those of existing approaches. We present the theoretical foundations of `PT-LiRPA`, along with a running example to illustrate our method for linearizing complex deep neural networks and practical implementation in Section 3, and 4, respectively. Finally, Section 5 reports on the extensive empirical evaluation of our approach on standard benchmarks in DNN verification.

2 Related Work and Preliminaries

2.1 Related Work

Formal Verification. In recent years, significant research has been dedicated to formal verification and especially to robustness verification (Liu et al. 2021; Wei et al. 2025). For example, (Belkhouja and Doppa 2022) shares with our work the goal of achieving certified robustness for neural networks, but differs significantly in both focus and methodology. Their approach is specifically designed for the time-series domain, leveraging statistical constraints and polynomial transformations to generate adversarial examples and derive robustness guarantees. In contrast, our work addresses general classification tasks and introduces a probabilistically sound verification method based on tight linear relaxation bounds. Other related work that integrates recurrent neural network(RNN)-based policy learning with formal verification is presented in (Carr et al. 2021). Specifically, the authors target policy

verification under temporal logic constraints in partially observable setups by extracting finite-state controllers from RNNs to enable model checking. In contrast, our work focuses on classification tasks and specifically robustness verification of general feedforward deep neural networks under input perturbations. For this type of verification is important to cite sound and complete verifiers such as mixed integer programming (MIP) (Tjeng et al. 2017) and satisfiability module theory(SMT)-based solvers (Katz et al. 2017; Wu et al. 2024). The most closely related works to our proposal comprise LiRPA-based verification approaches that focus on increasing the quality of linear bounds of the most popular activation functions, such as ReLU, and more general activation functions. More specifically, (Xu, Shi, et al. 2020) proposes a framework for deriving and computing near-optimal sound bounds with linear relaxation-based perturbation analysis for neural networks. This framework is the base of all the most famous state-of-the-art formal verification tools such as CROWN (H. Zhang, T.-W. Weng, et al. 2018), α-CROWN (Xu, H. Zhang, et al. 2021), β-CROWN (S. Wang, H. Zhang, et al. 2021), GCP-CROWN (H. Zhang, S. Wang, et al. 2022), the top performer on last years VNN-COMP (Brix et al. 2023; Müller et al. 2022). Notably, this approach has also been recently employed to both provably (Kotha et al. 2023) and approximate (X. Zhang et al. 2024) bounding neural network preimages.

All these approaches will be used as a worst-case verification result in our experiments.

Sampling-based Approaches for Provable Verification Certificates. Different approaches have tried to incorporate a sampling-based approach to enhance either the linear relaxation of arbitrary non-linear functions (Biktairov and Deshmukh 2023; Paulsen and C. Wang 2022; Xue et al. 2023) or the verification process (Balunovic et al. 2019), but still maintaining provable verification certificates. For instance, (Biktairov and Deshmukh 2023; Paulsen and C. Wang 2022) proposed a method synthesizing linear bounds for arbitrary complex activation functions, such as GeLU (Hendrycks and Gimpel 2016) and Swish (Ramachandran et al. 2017), by combining a sampling technique with an LP solver to synthesize candidate lower and upper bound coefficients and then certifying the final result via SMT solvers (Gao et al. 2013). In (Xue et al. 2023), the authors focus on estimating the actual domain of an activation function by combining Monte Carlo simulation and gradient descent methods to compute an underestimated domain, which is then paired with over-approximations to define provable linear bounds. If, on the one hand, our `PT-LiRPA` also employs a similar sampling-based procedure to compute the estimated domain of reachable sets, on the other hand, it differs fundamentally in both nature and focus. Specifically, our approach provides an explicit formula that, for any desired level of confidence, gives the number of samples sufficient to estimate the minimum and maximum pre-activation value of any node (over the set of inputs in C) with the given confidence. In addition, we are also able to estimate the maximum error between our estimates and the true extremal pre-activation values. Crucially, our approach adopts a probabilistic perspective to derive safety insights, whereas (Xue et al. 2023) relies on combining under- and over-approximations to produce provable bounds—an approach that may inherit the limitations of formal methods discussed earlier.

Probabilistic Verification. Recently, several works have explored probabilistic verification of machine learning models. For example, (Dvijotham et al. 2018; Morettin et al. 2024; Y. Zhang et al. 2025) focus on a probabilistic verification perspective for general machine learning models (e.g., deep generative/diffusion models), leveraging either uncertainty sources generated by specific encoders or using symbolic reasoning and probabilistic inference. (Webb et al. 2018) proposes a Monte Carlo-based method utilizing multi-level splitting to estimate the probability of rare events for robustness verification. Other approaches rely on Chernoff-Cramér bounds, e.g., (Couellan 2021) estimates the local variation of neural network mappings at training points to regularize the loss function. In (Mangal et al. 2019), the authors probabilistically certify a neural network by overapproximating input regions where robustness is violated. In (Pautov et al. 2022) a probabilistic certification approach is proposed that can be used in general attack settings to estimate the probability of a model failing if the attack is sampled from a certain distribution. Another line of work, like (T.-W. Weng et al. 2018) uses *extreme value theory* (EVT) (Haan and Ferreira 2006) to statistically estimate the local Lipschitz constant and assess the probabilistic robustness

of the model without relying on the overapproximation of LiRPA-based solvers. Finally, recent works (Berrada et al. 2021; Boetius et al. 2024; Marzari, Corsi, Cicalese, et al. 2023; Marzari, Corsi, Marchesini, Farinelli, and Cicalese 2024; Sivaramakrishnan et al. 2024) focus either on different types of verification, such as probabilistic enumeration of (un)safe region for neural networks, or on probabilistic specifications for robustness verification, which falls outside the scope of this paper, as we are interested in standard robustness verification with common specifications.

More closely related to our work are the approaches used in PROVEN (L. Weng, Chen, et al. 2019) and Randomized Smoothing (Cohen et al. 2019), which also focus on probabilistic robustness guarantees for neural network classifiers. PROVEN builds upon LiRPA-based techniques, combining them with concentration inequalities to derive probabilistic bounds on the network's output under input perturbations. Similarly, Randomized Smoothing constructs a smoothed classifier by adding random noise, typically Gaussian, to the input and then provides robustness guarantees by analyzing the output distribution of the smoothed model. `PT-LiRPA` aligns with these approaches in that it also leverages linear relaxation techniques and probabilistic reasoning, but introduces a novel perspective by focusing on how much sampling-based underestimations affect the intermediate linear bounds used during network linearization. This enables a finer-grained analysis of robustness with strong probabilistic guarantees, which is particularly useful in scenarios where the available methods for computing exact worst-case bounds either fail (by not terminating under reasonable time and space constraints) or terminate with only over-conservative bounds.

2.2 Notation and Problem Formulation

Consider a neural network classifier $f : \mathbb{R}^d \to \mathbb{R}$, where d is the input space dimension. Let N denote the number of layers. For each $i = 1, \ldots, N$, we let d_i be the number of nodes in layer i. We use $z_j^{(i)}$ to denote the jth node in layer i (according to some fixed ordering of the nodes in the same level). For a given input vector $\boldsymbol{x}$, we associate to node $z_j^{(i)}$ two values: the preactivation value, denoted by $z_j^{(i)}(\boldsymbol{x})$, and the postactivation value $\hat{z}_j^{(i)}(\boldsymbol{x})$ obtained by applying a (typically non-convex) activation function σ to the preactivation value, i.e., $\hat{z}_j^{(i)}(\boldsymbol{x}) = \sigma(z_j^{(i)}(\boldsymbol{x}))$. The preactivation value of node $z_j^{(i)}$ is obtained as a linear combination of the post-activation values of the nodes in the previous layer. In formulas, let $\boldsymbol{z}^{(i)}(\boldsymbol{x}) = (z_1^{(i)}(\boldsymbol{x}), \ldots, z_{d_i}^{(i)}(\boldsymbol{x}))$ and $\hat{\boldsymbol{z}}^{(i)}(\boldsymbol{x}) = (\hat{z}_1^{(i)}(\boldsymbol{x}), \ldots, \hat{z}_{d_i}^{(i)}(\boldsymbol{x}) = \sigma(\boldsymbol{z}^{(i)}(\boldsymbol{x})) = (\sigma(z_1^{(i)}(\boldsymbol{x})), \ldots, \sigma(z_{d_i}^{(i)}(\boldsymbol{x})))$. Then, $\boldsymbol{z}^{(i)}(\boldsymbol{x}) = \boldsymbol{W}^{(i)}\hat{\boldsymbol{z}}^{(i-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(i)}$, for some given inter level weight matrix $\boldsymbol{W}^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ and bias vector $\boldsymbol{b}^{(i)} \in \mathbb{R}^{d_i}$—as resulting from the network training.

As observed in the introduction, we assume, without loss of generality, that there is a single node in the Nth layer, which we simply denote by $z^{(N)}$.[1] Hence we have $f(\boldsymbol{x}) = \hat{\boldsymbol{z}}^{(N)}(\boldsymbol{x}) = \boldsymbol{z}^{(N)}(\boldsymbol{x})$.

In our following arguments, we assume that the activation function of each node is the ReLU, which is the most employed in the verification works of the literature (S. Wang, H. Zhang, et al. 2021; Xu, H. Zhang, et al. 2021), but the soundness of the proposed approach can also be extended for different non-linear scalar functions studied in the literature, such as Tanh, Sigmoid, GeLU, as in (Xu, Shi, et al. 2020). Hence, we define the robustness verification problem of deep neural networks as follows.

Given an input point of interest $\boldsymbol{x}_0$, for which $f(\boldsymbol{x}_0) > 0$, and an input perturbation region $C = C_{\boldsymbol{x}_0,\epsilon} = \{\boldsymbol{x} \mid ||\boldsymbol{x} - \boldsymbol{x}_0||_\infty \leq \epsilon\}$, i.e., we set $p = \infty$, obtaining an N-dimensional hypercube, we aim to find, if there exists, an input $\boldsymbol{x} \in C$ such that $f(\boldsymbol{x}) \leq 0$, thus resulting in a violation of the property. If $f(\boldsymbol{x}) > 0 \ \forall \boldsymbol{x} \in C$, we say f is robust (or verified) to all the possible input perturbations in C. Importantly, as we will show, `PT-LiRPA` can be seamlessly integrated into any state-of-the-art LiRPA method. Therefore, whenever the underlying

[1] One can simply enforce this condition for networks that do not satisfy this assumption by adding one layer and encoding, for instance, the robustness property that we aim to verify in a single output node as a margin between logits, which produces a positive output only if the correct label is predicted (Liu et al. 2021; S. Wang, H. Zhang, et al. 2021).

method supports the verification of specifications beyond the ℓ_∞ norm—such as non-convex specifications or related constraints—`PT-LiRPA` remains directly applicable. A possible way to prove the property is to solve the optimization problem in terms of $\min_{\boldsymbol{x}\in C} f(\boldsymbol{x})$ and by checking if the result is positive. Formally:

Definition 2.1 (*Robustness verification problem*)**.**
Input: A tuple $\mathcal{T} = \langle f, C\rangle$.
Output: `Robust` $\iff \min_{\boldsymbol{x}\in C} f(\boldsymbol{x}) := z^{(N)}(\boldsymbol{x}) > 0$.

Because of the effect of the activation functions σ applied to the value computed in each node, the resulting function f computed by the network is, in general, non-convex, making the above Robustness verification problem NP-hard (Katz et al. 2017). In order to cope with this issue, (in)complete verifiers usually relax the DNNs' non-convexity to obtain over-approximate sound lower and upper bounding functions on f, respectively, denoted by $\underline{f}$ and $\overline{f}$, i.e., $\underline{f}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq \overline{f}(\boldsymbol{x})$ for all $\boldsymbol{x} \in C$. Therefore, if $\underline{f}^* = \min_{\boldsymbol{x}\in C} \underline{f}(\boldsymbol{x}) > 0$, then also $f^* = \min_{\boldsymbol{x}\in C} f(\boldsymbol{x}) > 0$, i.e., the real minimum value of f will be positive, and similarly if $\overline{f}^* = \min_{\boldsymbol{x}\in C} \overline{f}(\boldsymbol{x}) \leq 0$ than also $f^* = \min_{\boldsymbol{x}\in C} f(\boldsymbol{x}) \leq 0$.

In both these situations, we can return a provable result. However, if $\underline{f}^* < 0 < \overline{f}^*$, we cannot be sure about the sign of f^*, and we typically have to proceed with a branch and bound (BaB) (**bab**) process. More specifically, many FV tools firstly recursively divide the original verification problem into smaller subdomains, either by dividing the perturbation region (S. Wang, Pei, et al. 2018) or by splitting ReLU neurons into positive/negative linear domains (Bunel et al. 2020). Secondly, they bound each subdomain with specialized (incomplete) verifiers, typically linear programming (LP) solvers (Ehlers 2017), which can fully encode neuron split constraints. The verification process ends once we verify all the subdomains of this searching tree, or we find a single counterexample $\boldsymbol{x}$ such that $\overline{f}(\boldsymbol{x}) \leq 0$. Even though LP-verifiers are mainly used in complete FV tools, recent LiRPA-based approaches (S. Wang, H. Zhang, et al. 2021; Xu, H. Zhang, et al. 2021) show how to solve an optimization problem that is equivalent to the costly LP-based methods with neuron split constraints while maintaining the efficiency of bound propagation techniques, significantly outperforming LP-verification time thanks to GPU acceleration.

2.3 Linear Relaxation-based Perturbation Analysis (LiRPA) Approaches

LiRPA approaches (Singh et al. 2019; S. Wang, H. Zhang, et al. 2021; Xu, Shi, et al. 2020; Xu, H. Zhang, et al. 2021; H. Zhang, T.-W. Weng, et al. 2018) propose to cope with the non-linearity of the function computed by a neural network by computing linear approximations of each non-linear unit. The high-level idea is to compute bounds on each neuron's function in the DNN, for instance, all the ReLU nodes, that can be expressed by linear functions, for which the above *robustness verification problem* can be efficiently solved. In detail, using interval bound propagation (IBP) (Lomuscio and Maganti 2017), we first compute a *reachable set* for each neuron $z_j^{(i)}$.

Definition 2.2 (Reachable set)**.** Given an input perturbation region $C \subseteq \mathbb{R}^d$, the reachable set of a neuron $z_j^{(i)}$ is defined as the interval $[l_j^{(i)}, u_j^{(i)}]$, where $l_j^{(i)} \leq z_j^{(i)}(\mathbf{x}) \leq u_j^{(i)} \quad$ for all $\mathbf{x} \in C$.

That is, $l_j^{(i)}$ and $u_j^{(i)}$ represent lower and upper bounds, respectively, on the pre-activation values of neuron $z_j^{(i)}$ over the entire perturbation region. The ReLU post-activation value of the node is given by $\hat{z}_j^{(i)} = \max(0, z_j^{(i)})$. Therefore, the node is considered *"unstable"* if its pre-activated bounds are such that $u_j^{(i)} > 0 > l_j^{(i)}$ and a linear approximate bound can be computed as depicted in Fig. 1. Once linear bounds are established across all neurons, two propagation methods are typically employed: forward and backward. In forward propagation, the linear bounds for each neuron are expressed in terms of the input and propagated layer by layer until the output is reached. In backward propagation, we start from the output and propagate the bounds backward to earlier layers

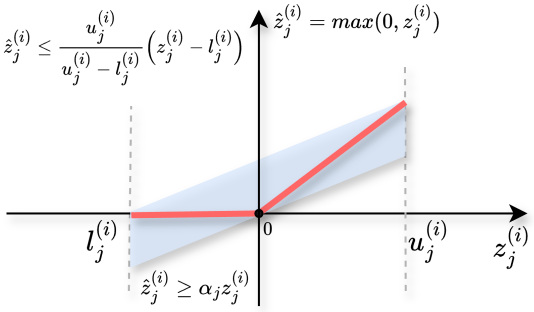

Fig. 1. Linear relaxation for ReLU($z_j^{(i)}$)

until we can express a linear relation between input and output. Otherwise, the node is either considered *"active"* if $l_j^{(i)} \geq 0$ or *"inactive"* if $u_j^{(i)} \leq 0$. In detail, to improve the tightness of the bounds, (Xu, Shi, et al. 2020) proposes a refined backward computation strategy that leverages information from a prior forward propagation. We report in Alg. 1 a brief overview of the method. After computing all reachable sets using a method such as interval bound propagation (Lomuscio and Maganti 2017), the backward analysis constructs layer-wise linear relaxations, starting from the output layer and working backward to the input. For clarity purposes, in the following, we will only report the notation for the linear lower bound computation, but similar considerations also apply to the upper bound.

For a DNN composed of N layers, the base case is defined as $\underline{\boldsymbol{A}}^{(N)} = I$. Moreover, in the case of a single output node, $\underline{\boldsymbol{A}}^{(N-1)} = \boldsymbol{w}^\top$, where $\boldsymbol{w}$ is the weight vector of the final layer. For the remaining layers $i \in \{N-2, \ldots, 1\}$, the linear relaxation is propagated using the recurrence $\underline{\boldsymbol{A}}^{(i)} = \underline{\boldsymbol{A}}^{(i+1)}\, \underline{\boldsymbol{D}}^{(i)}\, \boldsymbol{W}^{(i)}$, where $\boldsymbol{W}^{(i)}$ is the inter level weight matrix and $\underline{\boldsymbol{D}}^{(i)}$ is a diagonal matrix encoding the linear relaxation of the activation function at the i-th layer. For each layer i, we also recursively compute a bias vector $\underline{\boldsymbol{b}}^{(i)}$ based on the reachable sets of the i-th layer nodes and the matrix $\underline{\boldsymbol{A}}^{(i+1)}$. Each diagonal entry $\underline{\boldsymbol{D}}_{j,j}^{(i)}$ depends on the preactivation bounds of neuron j

Algorithm 1 `LiRPA`(H. Zhang, T.-W. Weng, et al. 2018) backward output bounds computation

Input: A DNN f with N layers, and input $\boldsymbol{x}_0$ and an ε perturbation to compute the perturbation region C, *interm_bounds* (optional).
Output: provable lower bound of f when considering C.

if *interm_bounds* $== \emptyset$ **then**
  interm_bounds $\leftarrow$ `IBP`(f, C)
$\underline{\boldsymbol{A}}^{(N)} = I$, $\underline{\boldsymbol{A}}^{(N-1)} = \boldsymbol{w}^\top$ ▷ $\boldsymbol{w}^\top$ is the weight vector of the final layer.
for $i \in \{N-2, \ldots, 1\}$ **do**
  $\underline{\boldsymbol{D}}^{(i)} \leftarrow$ `ComputeDiagonalMatrix`(interm_bounds$[i]$, $\underline{\boldsymbol{A}}^{(i+1)}$) ▷ as in Eq. 2
  $\underline{\boldsymbol{b}}^{(i)} \leftarrow$ `ComputeBiasVector`(interm_bounds$[i]$, $\underline{\boldsymbol{A}}^{(i+1)}$) ▷ as in Eq. 2
  $\underline{\boldsymbol{A}}^{(i)} \leftarrow \underline{\boldsymbol{A}}^{(i+1)}\, \underline{\boldsymbol{D}}^{(i)}\, \boldsymbol{W}^{(i)}$ ▷ $\boldsymbol{W}^{(i)}$ is the weight matrix of layer i.
$\underline{d} \leftarrow \underline{\boldsymbol{A}}^{(N-1)}\underline{\boldsymbol{b}}^{(N-2)} + \ldots + \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{b}}^{(1)}$
lower_bound $\leftarrow -||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \underline{\boldsymbol{A}}^{(1)}\boldsymbol{x}_0 + \underline{d}$ ▷ using Hölder's inequality for $\min_{\boldsymbol{x} \in C} \underline{\boldsymbol{A}}^{(1)}(\boldsymbol{x}) + \underline{d}$
return *lower_bound*

(computed during the forward pass) and the sign of $\underline{A}_j^{(i+1)}$, i.e., the coefficient associated with neuron j in the linear relaxation of the next layer, for which we report the explicit formulas used for computing $\underline{D}^{(i)}$ and $\boldsymbol{b}^{(i)}$ in the example provided below. At the end of the process, a provable lower bound on the minimum of f in C (the ℓ_∞ norm ball around $\boldsymbol{x}_0$) is then easily obtained using Hölder's inequality (H. Zhang, T.-W. Weng, et al. 2018) as

$$\min_{\boldsymbol{x}\in C} \boldsymbol{a}^T_{\text{LiRPA}}(\boldsymbol{x}) + \boldsymbol{c}_{\text{LiRPA}} = -||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \underline{\boldsymbol{A}}^{(1)}\boldsymbol{x} + \boldsymbol{c}_{\text{LiRPA}}. \tag{1}$$

with $\boldsymbol{c}_{\text{LiRPA}} = \underline{\boldsymbol{A}}^{(N)}\underline{\boldsymbol{b}}^{(N-1)} + \ldots + \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{b}}^{(1)}$.

To provide the reader a concrete and practical illustration of this approach, in the following, we present a simple example of linear bound computation for the toy DNN shown in Figure 2, using CROWN, a state-of-the-art LiRPA-based method (H. Zhang, T.-W. Weng, et al. 2018).

Example of linear computation with LiRPA. Consider a neural network with two inputs, two hidden layers with ReLU activation, and one single output. Following the notation introduced in Sec. 2 we define:

$$\boldsymbol{W}^{(1)} = \begin{bmatrix} 2 & 1 \\ -3 & 4 \end{bmatrix}, \quad \boldsymbol{W}^{(2)} = \begin{bmatrix} 4 & -2 \\ 2 & 1 \end{bmatrix}, \quad \boldsymbol{w}^{(3)T} = [-2, 1];$$

and, for simplicity, we set the bias terms in the layers to zero. We consider an original input $\boldsymbol{x}_0{}^T = [0, 1]$ and an ℓ_∞ $\varepsilon = 2$ perturbation around it, thus obtaining a perturbation region $C = [[-2, 2], [-1, 3]]$.

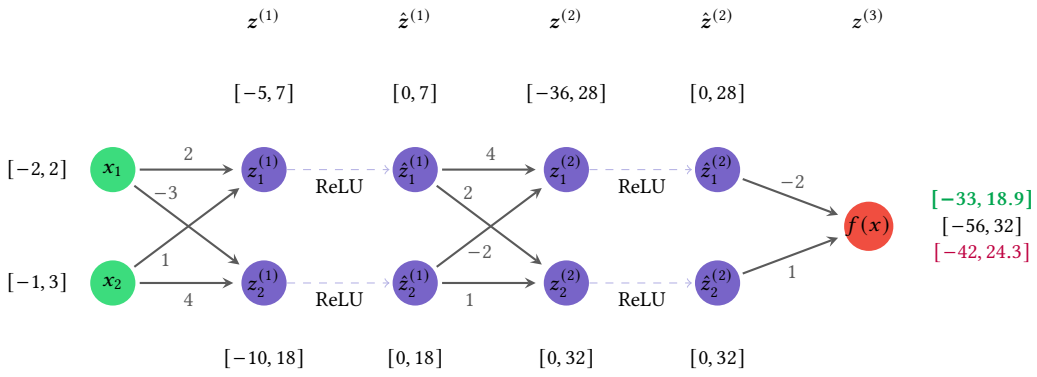

Fig. 2. Toy DNN used in this example. Intervals reported in green are the exact output reachable set computed via MIP, in black are the ones of IBP, and finally, in purple, the results for CROWN considering the input $[[-2, 2], [-1, 3]]$.

By propagating these intervals through the DNN, we obtain the interval $[-56, 32]$ as the output reachable set. Given the reasonable size of the neural network, before computing the linear lower and upper bounds using `LiRPA`, we employed the sound and complete MIP (Tjeng et al. 2017) solver to compute the true min and max of the function, respectively, which correspond to $[-33, 18.89]$, highlighted in green in Fig. 2.

To compute the lower and upper bounds using CROWN (H. Zhang, T.-W. Weng, et al. 2018), we employ LiRPA's backward computation strategy. To this end, it is helpful to represent the neural network as reported in Fig. 3.

We note that $\hat{\boldsymbol{z}}^{(2)}$ and $\hat{\boldsymbol{z}}^{(1)}$ contain non-linear activation functions (ReLU), and we have to linearize them to keep the linear relationship between the output and these hidden layers. To this end, we create $2 \times$ #ReLU layers (for the lower and upper bound, respectively) diagonal matrices $\underline{\boldsymbol{D}}^{(2)}, \overline{\boldsymbol{D}}^{(2)}, \underline{\boldsymbol{D}}^{(1)}, \overline{\boldsymbol{D}}^{(1)}$ and bias vectors $\underline{\boldsymbol{b}}^{(2)}, \overline{\boldsymbol{b}}^{(2)}, \underline{\boldsymbol{b}}^{(1)}, \overline{\boldsymbol{b}}^{(1)}$ reflecting the impact of each ReLU nodes on the final output. We report for simplicity here

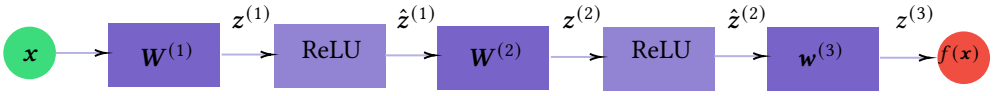

Fig. 3. Alternative representation of toy DNN of Figure 2.

the original definition provided in (H. Zhang, T.-W. Weng, et al. 2018) (a similar definition is applied to compute the i-th layer $\overline{D}^{(i)}$ and $\overline{b}^{(i)}$):

$$\underline{D}^{(i)} = \begin{cases} 1 & l_j \geq 0, \\ 0 & u_j \leq 0, \\ \alpha_j & u_j > 0 > l_j \text{ and } A_j^{(i+1)} \geq 0, \\ \frac{u_j}{u_j - l_j} & u_j > 0 > l_j \text{ and } A_j^{(i+1)} < 0 \end{cases} \qquad \underline{b}^{(i)} = \begin{cases} 0 & l_j > 0 \text{ or } u_j \leq 0, \\ 0 & u_j > 0 > l_j \text{ and } A_j^{(i+1)} \geq 0, \\ -\frac{u_j l_j}{u_j - l_j} & u_j > 0 > l_j \text{ and } A_j^{(i+1)} < 0. \end{cases} \tag{2}$$

In the following, for simplicity, we always set $\alpha_j = 0$. After defining the i-th diagonal matrix, we can compute the i-th layer relaxation with respect to the output as $\underline{A}^{(i)} = \underline{A}^{(i+1)}\underline{D}^{(i)}W^{(i)}$ and similarly for the $\overline{A}^{(i)}$. In the beginning, we set $\underline{A}^{(4)} = \overline{A}^{(4)} = I$ and $\underline{A}^{(3)} = \overline{A}^{(3)} = {w^{(3)}}^T$ and write starting from right to left (backward computation)[2]

$$\begin{aligned} f(x) &= z^{(3)}(x) \\ &= {w^{(3)}}^T \hat{z}^{(2)}(x) \\ &\geq \underline{A}^{(3)}\underline{D}^{(2)} z^{(2)}(x) && \text{computing a linearization for } \hat{z}^{(2)} \\ &\geq \underbrace{\underline{A}^{(3)}\underline{D}^{(2)}W^{(2)}}_{\underline{A}^{(2)}} \hat{z}^{(1)}(x) && \text{rewriting } z^{(2)} = W^{(2)}\hat{z}^{(1)} \\ &\geq \underline{A}^{(2)}\underline{D}^{(1)} z^{(1)}(x) && \text{computing a linear bound for } \hat{z}^{(1)} \\ &\geq \underbrace{\underline{A}^{(2)}\underline{D}^{(1)}W^{(1)}}_{\underline{A}^{(1)}}(x) && \text{rewriting } z^{(1)} = W^{(1)}\hat{z}^{(0)} = W^{(1)}(x) \\ &\geq \underline{A}^{(1)}(x) + \underline{d}. \end{aligned}$$

Hence, in order to linearize $\hat{z}^{(2)}(x)$ we compute $\underline{D}^{(2)}, \overline{D}^{(2)}$ and $\underline{b}^{(2)}, \overline{b}^{(2)}$ which presely correspond to

$$\underline{D}^{(2)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.4375 & 0 \\ 0 & 1 \end{bmatrix} \qquad \underline{b}^{(2)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 15.75 \\ 0 \end{bmatrix}$$

$$\overline{D}^{(2)} = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \overline{b}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

[2]We report the lower bound version but for the upper we have similar consideration with the reversed inequality.

where $\underline{D}^{(2)}_{j,j}$ element is computed looking at each intermediate pre-activated bounds of $z^{(2)}_j$ and the sign of j-th element of the vector $\underline{A}^{(3)}$. Thus we have $\underline{A}^{(2)} = \underline{A}^{(3)}\underline{D}^{(2)}W^{(2)} = [-1.5, 2.75]$ and $\overline{A}^{(2)} = \overline{A}^{(3)}\overline{D}^{(2)}W^{(2)} = [2, 1]$. We proceed computing the diagonal matrix $\underline{D}^{(1)}$, $\overline{D}^{(1)}$ and bias vectors $\underline{b}^{(1)}, \overline{b}^{(1)}$ for $\hat{z}^{(1)}$. In detail, we obtain,

$$\underline{D}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \alpha \end{bmatrix} = \begin{bmatrix} 0.583 & 0 \\ 0 & 0 \end{bmatrix} \qquad \underline{b}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 2.92 \\ 0 \end{bmatrix}$$

$$\overline{D}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \frac{u}{u-l} \end{bmatrix} = \begin{bmatrix} 0.583 & 0 \\ 0 & 0.643 \end{bmatrix} \qquad \overline{b}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ \frac{-ul}{u-l} \end{bmatrix} = \begin{bmatrix} 2.92 \\ 6.43 \end{bmatrix}$$

with $\underline{A}^{(1)} = \underline{A}^{(2)}\underline{D}^{(1)}W^{(1)} = [-1.75, -0.875]$ and $\overline{A}^{(1)} = \overline{A}^{(2)}\overline{D}^{(1)}W^{(1)} = [0.40, 3.74]$.

Finally, we compute the sum in the bias vectors $\underline{d} = \underline{A}^{(3)}\underline{b}^{(2)} + \underline{A}^{(2)}\underline{b}^{(1)} = -35.88$ and $\overline{d} = \overline{A}^{(3)}\overline{b}^{(2)} + \overline{A}^{(2)}\overline{b}^{(1)} = 12.27$. The final linear relation is thus $\underline{f}(\boldsymbol{x}) \geq \underline{A}^{(1)}(\boldsymbol{x}) + \underline{d}$ and $\overline{f}(\boldsymbol{x}) \leq \overline{A}^{(1)}(\boldsymbol{x}) + \overline{d}$. Using Hölder's inequality (H. Zhang, T.-W. Weng, et al. 2018), we obtain

$$\begin{aligned} \underline{f}_{\text{CROWN}} = \min_{\boldsymbol{x}\in C} \underline{A}^{(1)}(\boldsymbol{x}) + \underline{d} &= -||\underline{A}^{(1)}||_1 \cdot \varepsilon + \underline{A}^{(1)}\boldsymbol{x}_0 + \underline{d} \\ &= -5.25 - 0.875 - 35.88 = -42. \end{aligned}$$

$$\begin{aligned} \overline{f}_{\text{CROWN}} = \max_{\boldsymbol{x}\in C} \overline{A}^{(1)}(\boldsymbol{x}) + \overline{d} &= ||\underline{A}^{(1)}||_1 \cdot \varepsilon + \overline{A}^{(1)}\boldsymbol{x}_0 + \overline{d} \\ &= 8.28 + 3.74 + 12.27 = 24.3. \end{aligned}$$

As we can notice, we obtain a tight over-approximation of the true lower and upper bounds, significantly improving the bounds derived with naive IBP ([-56,32]).

3 Probabilistically Tightened LiRPA via Underestimation

In this section, we theoretically investigate whether and how it is possible to compute tight intermediate reachable sets, which directly impact the linear output bound computation. As highlighted in Alg. 3, the entire linearization process crucially depends on the bounds computed at line 3. However, computing exact values for such bounds is generally infeasible, as the problem has been shown to be NP-hard (Katz et al. 2017; L. Weng, H. Zhang, et al. 2018). Motivated by the speculation of Xu, H. Zhang, et al. (2021), we therefore explore efficient alternatives for approximating the exact (unknown) values of these intermediate bounds as tightly as possible. To this end, we study the impact of a sampling-based approach and what type of probabilistic guarantees can be achieved with it. In detail, we begin employing the *statistical prediction of tolerance limits* results (Wilks 1942), which allows a closed-form derivation of the required sample size to achieve a desired confidence level. However, as we will show, this method only quantifies the fraction of the perturbation region where the guarantees may fail, without addressing the magnitude of potential violations relative to the estimated bounds. Consequently, the resulting probabilistic certificates are weaker than those provided by related approaches such as (Cohen et al. 2019; L. Weng, Chen, et al. 2019), where guarantees hold across the entire perturbation region. To mitigate this limitation, we first introduce a qualitative extension of Wilks' guarantees adapted to our setting. While novel, this bound can become overly loose in high-dimensional scenarios. To address this issue, we then develop a new theoretical and practical result based on *extreme value theory* (Haan and Ferreira 2006), which allows us to tightly bound the magnitude of possible violations with respect to the estimated bounds.

We now present all the theoretical and practical components to compute tightened bounds with probabilistic guarantees within our `PT-LiRPA` approach. Our approach is based on a statistical methodology that allows us to compute estimates on the neural network's output in the form $\min_{\boldsymbol{x}\in C} f(\boldsymbol{x})$ and $\max_{\boldsymbol{x}\in C} f(\boldsymbol{x})$, which hold with

high confidence. In particular, we employ the tools of *statistical prediction of tolerance limits* of (Wilks 1942). Fix a node $z_j^{(i)}$ in the neural network and, as before, let $z_j^{(i)}(\boldsymbol{x})$ be the preactivation value for the node when the input to the network is a vector $\boldsymbol{x}$. Let $\boldsymbol{x}_1, \ldots \boldsymbol{x}_n$ be n points (vectors) independently and uniformly sampled from the perturbation set of interest C. We compute $z_j^{(i)}$'s (estimated) pre-activated bounds as:

$$\bar{l}_j^{(i)} = \min_{k=1,\ldots,n} z_j^{(i)}(\boldsymbol{x}_k); \qquad \underline{u}_j^{(i)} = \max_{k=1,\ldots,n} z_j^{(i)}(\boldsymbol{x}_k),$$

i.e., the minimum and maximum value observed from the propagation of the n random points in that specific neuron $z_j^{(i)}$. Let $l_j^{*(i)} = \min_{\boldsymbol{x}\in C} z_j^{(i)}(\boldsymbol{x})$ and $u_j^{*(i)} = \max_{\boldsymbol{x}\in C} z_j^{(i)}(\boldsymbol{x})$. Clearly, we have

$$\bar{l}_j^{(i)} \geq l_j^{*(i)}; \quad \underline{u}_j^{(i)} \leq u_j^{*(i)}. \tag{3}$$

However, based on the results of (Wilks 1942), for any $\psi, R \in (0,1)$, we can choose the sample size n that guarantees that the estimated reachable set is correct with probability ψ for at least a fraction R of points in the perturbation set C. Crucially, this statistical result does not require any knowledge of the probability distribution governing the output of our function of interest. Formally, we have the following.

Lemma 3.1 (Probabilistically tightened reachable sets)**.** *Let n the number of samples employed in the computation and the interval $[\bar{l}_j^{(i)}, \underline{u}_j^{(i)}]$, where $\bar{l}_j^{(i)}$ and $\underline{u}_j^{(i)}$ are the minimum and maximum pre-activation values observed in the sample, respectively. Fix $R \in (0,1)$, then for any further possibly infinite sequence of samples from C, the probability that $[\bar{l}_j^{(i)}, \underline{u}_j^{(i)}]$ is correct[3] for at least a fraction R of points is at least $\psi = n \cdot \int_R^1 x^{n-1}\, dx = (1 - R^n)$.*

Hence, following Lemma 3.1, whose proof follows from (Wilks 1942), for any desired confidence level ψ, and lower bound fraction R, we can compute the number n of samples sufficient to obtain the provable probabilistic guarantees on the desired reachable set accuracy. Specifically, we have that if we use a sample of $n \geq \frac{\ln(1-\psi)}{\ln(R)}$ input points and with them we obtain an estimated reachable set $[\bar{l}_j^{(i)}, \underline{u}_j^{(i)}]$, then with probability ψ at most a fraction $(1-R)$ of points in an indefinitely larger future sample could fall outside such reachable set. By taking into account the total number of neurons in the DNN, and using a independently chosen set of points for each neuron, we can extend the above result to compute reachable sets for each neuron that are simultaneously correct with probability ψ for at least a fraction R of the points in C:

Proposition 3.2. *Consider an N-layer ReLU DNN with m neurons. Fix a confidence level ψ and coverage ratio $R \in (0,1)$. For each intermediate neuron $z_j^{(i)}$, collect n' i.i.d. samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ from the perturbation region C, and compute the approximate reachable set as $\bar{l}_j^{(i)} = \min_{k=1,\ldots,n'} z_j^{(i)}(\boldsymbol{x}_k)$ and $\underline{u}_j^{(i)} = \max_{k=1,\ldots,n'} z_j^{(i)}(\boldsymbol{x}_k)$. If the number of samples used for each neuron satisfies $n' \geq \frac{\ln(1-\psi^{1/m})}{\ln(1-(1-R)/m)}$, then for any (possibly infinite) sequence X of inputs sampled independently and uniformly from C, for each neuron $z_j^{(i)}$ with probability at least $\psi^{1/m}$ there is a subsequence X' of X of size $|X'| \geq \left(1 - \frac{1-R}{m}\right)|X|$ such that for each $\boldsymbol{x} \in X'$ the estimated reachable set is sound, i.e., $z_j^{(i)}(\boldsymbol{x}) \in [\bar{l}_j^{(i)}, \underline{u}_j^{(i)}]$.*

We will now show that we can combine the (probabilistically valid) estimation computed on the reachable sets of individual neurons using a LiRPA approach, so as to obtain, first, a probabilistically valid over-approximation of any ReLU layer in the DNN and thus on the final network's lower and upper outputs. In detail, we begin by proving that the estimated reachable sets used to produce the vectors $\boldsymbol{A}$ and $\boldsymbol{b}$, together with the linearization applied to each ReLU layer of the network (as in Sec. 2.3), yield a probabilistically sound over-approximation that covers at least a fraction R of the perturbation region C.

[3]In the sense that there exists $C' \subseteq C$ such that $|C'|/|C| \geq R$ and for all $\boldsymbol{x} \in C'$ it holds that $z_j^{(i)}(\boldsymbol{x}) \in [\bar{l}_j^{(i)}, \underline{u}_j^{(i)}]$.

Lemma 3.3 (ReLU Layer Relaxation using PT-LiRPA)**.** *Consider a ReLU DNN with m neurons distributed over N layers, and C a perturbation region of interest. Fix confidence and coverage parameter $\psi, R \in (0,1)$. Fix a layer $i \in \{1,\dots,N-2\}$, and for each $j = 1,\dots,d_i$ compute an estimated reachable set $[\bar{l}_j, \underline{u}_j]$ for neuron $z_j^{(i)}$, using $n' \geq m\frac{ln(1-\psi^{1/m})}{ln(1-(1-R)/m)}$ independently and uniformly sampled point from C, as by Proposition 3.2. Let $\bar{\boldsymbol{l}} = (\bar{l}_1,\dots,\bar{l}_{d_i})$ and $\underline{\boldsymbol{u}} = (\underline{u}_1,\dots,\underline{u}_{d_i})$. Let $\boldsymbol{A}, \boldsymbol{b}$ be the vectors of the linear bounds coefficients and biases (inductively) computed for the ReLU layer $i+1$, and such that, with probability $\geq \psi^{\frac{d_{i+1}+d_{i+2}+\cdots+d_N}{m}}$, for any $n \in \mathbb{N}$ in any sequence of n points uniformly and independently sampled from C, for at least $n \times \left(1 - \frac{1-R}{m}\sum_{j=i+1}^{N} d_j\right)$ points $\boldsymbol{x}$ in X it holds that*

$$f(\boldsymbol{x}) \geq \boldsymbol{A}^T ReLU(\boldsymbol{v_x}) + \boldsymbol{b}, \tag{4}$$

where $\boldsymbol{v_x}$ is the vector of pre-activation values of the neurons in the layer i when the input is $\boldsymbol{x}$. Then,

(1) *with probability $\geq \psi^{\frac{d_i}{m}}$, it holds that for any $n \in \mathbb{N}$ in any sequence X of n points uniformly and independently sampled from C, for at least $n \times \left(1 - \frac{1-R}{m} d_i\right)$ points $\boldsymbol{x}$ in X it holds that:*

$$\boldsymbol{A}^T ReLU(\boldsymbol{v_x}) \geq \boldsymbol{A}^T(\underline{\boldsymbol{D}}^* \boldsymbol{v_x} + \underline{\boldsymbol{b}}^*) \tag{5}$$

where

$$\underline{\boldsymbol{D}}^* = \begin{cases} 1 & \bar{\boldsymbol{l}}_j \geq 0, \\ 0 & \underline{\boldsymbol{u}}_j \leq 0, \\ \alpha_j & \underline{\boldsymbol{u}}_j > 0 > \bar{\boldsymbol{l}}_j \text{ and } \boldsymbol{A}_j \geq 0, \\ \frac{\underline{\boldsymbol{u}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j} & \underline{\boldsymbol{u}}_j > 0 > \bar{\boldsymbol{l}}_j \text{ and } \boldsymbol{A}_j < 0 \end{cases} \quad \underline{\boldsymbol{b}}^* = \begin{cases} 0 & \bar{\boldsymbol{l}}_j > 0 \text{ or } \underline{\boldsymbol{u}}_j \leq 0, \\ 0 & \underline{\boldsymbol{u}}_j > 0 > \bar{\boldsymbol{l}}_j \text{ and } \boldsymbol{A}_j \geq 0, \\ -\frac{\underline{\boldsymbol{u}}_j \bar{\boldsymbol{l}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j} & \underline{\boldsymbol{u}}_j > 0 > \bar{\boldsymbol{l}}_j \text{ and } \boldsymbol{A}_j < 0. \end{cases}$$

(2) *with probability $\geq \psi^{\frac{d_i+d_{i+1}+\cdots+d_N}{m}}$, for vectors $\boldsymbol{A}' = \boldsymbol{A}^T\boldsymbol{D}^*$ and $\boldsymbol{b}' = \boldsymbol{b} + \boldsymbol{A}^T\boldsymbol{b}^*$ it holds that for any $n \in \mathbb{N}$ in any sequence X of n points uniformly and independently sampled from C, for at least $n \times \left(1 - \frac{1-R}{m}\sum_{j=i}^{N} d_j\right)$ points $\boldsymbol{x}$ in X we have that*

$$f(\boldsymbol{x}) \geq \boldsymbol{A}' ReLU(\boldsymbol{v_x}) + \boldsymbol{b}', \tag{6}$$

Proof. The proof closely follows the analogous result at the basis of the LiRPA approach of H. Zhang, T.-W. Weng, et al. (2018). The only (crucial) difference is that we construct $\underline{\boldsymbol{D}}^*$ and $\underline{\boldsymbol{b}}^*$, i.e., the diagonal matrix, and the bias vector meant to provide a linear lower bound on the post-activation values of layer i, using the vectors of estimated reachable sets of the nodes in the layer, instead of their actual reachable sets.

Let X be a sequence of n points independently and uniformly sampled from C.

For each $j = 1,\dots,d_i$, let

$$X_j = \{\boldsymbol{x} \in X \mid z_j^{(i)}(\boldsymbol{x}) \notin [\bar{l}_j, \underline{u}_j]\},$$

and let $\mathcal{E}_j$ be the event

$$\mathcal{E}_j = \{|X_j| \leq \frac{1-R}{m}|X|\}.$$

Because of the way we compute the estimated reachable sets $[\bar{l}_j, \underline{u}_j]$ (Proposition 3.2) we have that for each j

$$Pr[\mathcal{E}_j] \geq \psi^{1/m}.$$

Moreover, these events are independent, hence

$$Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \cdots \wedge \mathcal{E}_{d_i}] \geq \psi^{d_i/m}.$$

Let

$$X_{OK} = X \setminus \left(\bigcup_{j=1}^{d_i} X_j \right) = \{ \boldsymbol{x} \in X \mid \forall j, z_j^{(i)} \in [\bar{l}_j, \underline{u}_j] \}.$$

Then,

$$|X_{OK}| \geq |X| - \sum_{j=1}^{d_i} |X_j|,$$

and in particular, if for each j it holds that $|X_j| \leq \frac{1-R}{m}|X|$–i.e., when the event $\mathcal{E}_1 \wedge \cdots \wedge \mathcal{E}_{d_i}$ occurs–we have $|X_{OK}| \geq \left(1 - \frac{(1-R)}{m} d_i\right)|X|$.

We can conclude that the probability of the event $\mathcal{E}_{OK} = \{|X_{OK}| \geq (1 - \frac{(1-R)d_i}{m})|X|\}$ satisfies

$$Pr[\mathcal{E}_{OK}] \geq Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \cdots \wedge \mathcal{E}_{d_i}] \geq \psi^{d_i/m}.$$

In words, we have shown that with probability $\psi^{1/m}$ for each point $\boldsymbol{x}$ in a fraction of X of size $(1 - \frac{(1-R)d_i}{m})|X|$, for all nodes $z_j^{(i)}$ in layer i, the pre-activation value $z_j^{(i)}(\boldsymbol{x})$ induced by $\boldsymbol{x}$ is contained in the estimated reachable sets $[\bar{l}_j, \underline{u}_j]$.

In order to simplify the notation, let us fix an $\boldsymbol{x}$ from X_{OK} and let $\boldsymbol{v} = \boldsymbol{z}^{(i)}(\boldsymbol{x})$ be the vector of pre-activation values induced by $\boldsymbol{x}$ in the nodes of layer i. We will show that for the $\underline{\boldsymbol{D}}^*$ and $\underline{\boldsymbol{b}}^*$ defined in the statement, we have that for each j it holds that $\boldsymbol{A}_j(ReLU(\boldsymbol{v}_j)) \geq \boldsymbol{A}_j(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j)$, which immediately implies the desired result, since $\boldsymbol{A}^T ReLU(\boldsymbol{v}) = \sum_j \boldsymbol{A}_j(ReLU(\boldsymbol{v}_j)) \geq \boldsymbol{A}_j(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j) = \boldsymbol{A}^T(\underline{\boldsymbol{D}}^*\boldsymbol{v}_{\boldsymbol{x}} + \underline{\boldsymbol{b}}^*)$.

We start by noticing that the following inequalities hold

$$\boldsymbol{l}_j \leq \boldsymbol{l}_j^* \leq \bar{\boldsymbol{l}}_j < 0 < \underline{\boldsymbol{u}}_j \leq \boldsymbol{u}_j^* \leq \boldsymbol{u}_j, \tag{7}$$

where $[\boldsymbol{l}_j^*, \boldsymbol{u}_j^*]$ is the actual reachable set for the j-th node of the layer under consideration.

We split the argument into three cases according to the sign of the estimated values $\bar{l}_j$ and $\underline{u}_j$. Moreover, we split each case into subcases according to the sign of $\boldsymbol{A}_j$ and $\boldsymbol{v}_j$.

Case 1. $\underline{\boldsymbol{u}}_j > 0 > \bar{\boldsymbol{l}}_j$

From inequality 7, we know that since we are underestimating true bounds $[\boldsymbol{l}_j^*, \boldsymbol{u}_j^*]$, the ReLU node is actually unstable, even for any LiRPA approach. Comparing the diagonal coefficients $\frac{\underline{\boldsymbol{u}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}$ with $\frac{\boldsymbol{u}_j}{\boldsymbol{u}_j - \boldsymbol{l}_j}$ and the biases $-\frac{\underline{\boldsymbol{u}}_j \bar{\boldsymbol{l}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}$ with $-\frac{\boldsymbol{u}_j \boldsymbol{l}_j}{\boldsymbol{u}_j - \boldsymbol{l}_j}$ of `PT-LiRPA` and any LiRPA cannot be helpful. The relation between the coefficients strongly depends on the quality of the bounds computed, and we cannot draw any direct conclusion since in some cases $\underline{\boldsymbol{D}} > \underline{\boldsymbol{D}}^*$ and in some cases not. Hence, we need to proceed by subcases.

Subcase 1.1. $\boldsymbol{A}_j < 0$ and $\boldsymbol{v}_j < 0$.

Since $\boldsymbol{v}_j < 0$ then $ReLU(\boldsymbol{v}_j) = 0$. Moreover, we have

$$(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j) = \frac{\underline{\boldsymbol{u}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}\boldsymbol{v}_j + \left(- \frac{\underline{\boldsymbol{u}}_j \bar{\boldsymbol{l}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j} \right) = \frac{\underline{\boldsymbol{u}}_j(\boldsymbol{v}_j - \bar{\boldsymbol{l}}_j)}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j} \geq 0,$$

where the last inequality holds since $\boldsymbol{v}_j \in [\bar{\boldsymbol{l}}_j, \underline{\boldsymbol{u}}_j]$ implies $\boldsymbol{v}_j - \bar{\boldsymbol{l}}_j \geq 0$. Multiplying both sides by $\boldsymbol{A}_j < 0$ and using $ReLU(\boldsymbol{v}_j) = 0$, we get

$$\boldsymbol{A}_j(ReLU(\boldsymbol{v}_j)) = 0 \geq \boldsymbol{A}_j(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j)$$

Subcase 1.2. $\boldsymbol{A}_j < 0$ and $\boldsymbol{v}_j \geq 0$.
Since $\boldsymbol{v}_j \geq 0$ then $ReLU(\boldsymbol{v}_j) = \boldsymbol{v}_j$. Moreover, we have

$$(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j) = \frac{\underline{\boldsymbol{u}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}\boldsymbol{v}_j + \left(-\frac{\underline{\boldsymbol{u}}_j\bar{\boldsymbol{l}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}\right) \geq \boldsymbol{v}_j,$$

where the last inequality holds since under the standing hypothesis the coefficient of $\boldsymbol{v}_j$ in the left hand side is ≥ 1 and term $-\frac{\underline{\boldsymbol{u}}_j\bar{\boldsymbol{l}}_j}{\underline{\boldsymbol{u}}_j - \bar{\boldsymbol{l}}_j}$ is non-negative. Multiplying both sides by $\boldsymbol{A}_j < 0$ and using $ReLU(\boldsymbol{v}_j) = \boldsymbol{v}_j$, we get

$$\boldsymbol{A}_j(ReLU(\boldsymbol{v}_j)) = \boldsymbol{A}_j\boldsymbol{v}_j \geq \boldsymbol{A}_j(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j).$$

Subcase 1.3. $\boldsymbol{A}_j \geq 0$ and $\boldsymbol{v}_j < 0$.
We have

$$ReLU(\boldsymbol{v}_j) = 0 \geq \alpha_j\boldsymbol{v}_j + 0 = \underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j\,,$$

since $0 < \alpha_j$. Hence, multiplying both sides by $\boldsymbol{A}_j \geq 0$ we obtain again the desired inequality. Subcase 1.4. $\boldsymbol{A}_j \geq 0$ and $\boldsymbol{v}_j \geq 0$.
We have

$$ReLU(\boldsymbol{v}_j) = \boldsymbol{v}_j \geq \alpha_j\boldsymbol{v}_j + 0 = \underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j$$

since $\alpha < 1$. Again, multiplying both sides by $\boldsymbol{A}_j \geq 0$ we obtain again the desired inequality. This concludes the first case.

Case 2. $\bar{\boldsymbol{l}}_j > 0$

Since $\boldsymbol{v}_j \in [\bar{l}_j, \underline{\boldsymbol{u}}_j]$, with $\bar{l}_j > 0$ we have $ReLU(\boldsymbol{v}_j) = \boldsymbol{v}_j$.
Moreover, we have

$$\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j = 1 \cdot \boldsymbol{v}_j + 0 = \boldsymbol{v}_j = ReLU(\boldsymbol{v}_j)$$

from which, multiplying both sides by $\boldsymbol{A}_j$ yields the desired inequality $\boldsymbol{A}_j ReLU(\boldsymbol{v}_j) \geq \boldsymbol{A}_j(\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j)$.

Case 3. $\underline{\boldsymbol{u}}_j < 0$

Since $\boldsymbol{v}_j \in [\bar{l}_j, \underline{\boldsymbol{u}}_j]$ and $\underline{\boldsymbol{u}}_j < 0$ we have $ReLU(\boldsymbol{v}_j) = 0$. Moreover, under the standing hypothesis, we also have $\underline{\boldsymbol{D}}^*_{j,j}\boldsymbol{v}_j + \underline{\boldsymbol{b}}^*_j = 0 = ReLU(\boldsymbol{v}_j)$. Again, multiplying both sides by $\boldsymbol{A}_j$, we have that the desired inequality holds also in this case. We have proved that for each $\boldsymbol{x}$ in X_{OK} we have $\boldsymbol{A}^T ReLU(\boldsymbol{v}_j) \geq \boldsymbol{A}^T(\underline{\boldsymbol{D}}^*\boldsymbol{v}_x + \underline{\boldsymbol{b}}^*)$. Finally, recalling that with probability $\psi^{d_i/m}$ we have that $|X_{OK}| \geq |X|(1 - \frac{1-R}{m}d_i)$, we have that the previous inequality holds with the desired statistical guarantees, which completes the proof of claim (1).

For proving claim (2) it is enough to note that the guarantees on $\boldsymbol{A}, \boldsymbol{b}$ and (4) hold independently of the choice of the samples leading to the statistical guarantee on (5). Hence they simultaneously happen–yielding that (6) holds–with the product of the probabilities of (4) and (5), i.e., $\psi^{\frac{d_i+d_{i+1}+\cdots+d_N}{m}}$. In particular, with this probability, for any $n \in \mathbb{N}$ in any sequence X of n points uniformly and independently sampled from C, neither of (4) and (5) fails on at least $n \times \left(1 - \frac{1-R}{m}\sum_{j=i}^{N} d_j\right)$ points.

□

As a direct implication of this result, we can show that the neural network's output linear bounds computed using PT-LiRPA remain, with high probability, an overestimation of the real lower and upper bound, respectively, for at least a fixed fraction R of the perturbation region under consideration.

Theorem 3.4 (PT-LiRPA weak probabilistic guarantees)**.** *Fix an N-layer ReLU DNN with m neurons with f the function it computes. Then, for any $\psi, R \in (0,1)$ PT-LiRPA, using a total number of $n \geq m\frac{\ln(1-\psi^{1/m})}{\ln(1-(1-R)/m)}$ independent and uniformly distributed input points, computes a linear approximation $\boldsymbol{a}^T_{\texttt{PT-LiRPA}}(\boldsymbol{x}) + \boldsymbol{c}_{\texttt{PT-LiRPA}}$ of f such that with probability at least ψ for at least a fraction R of possibly infinite samples of input points, $\boldsymbol{x}$, it holds that*

$$f(\boldsymbol{x}) \geq \boldsymbol{a}^T_{\texttt{PT-LiRPA}}(\boldsymbol{x}) + \boldsymbol{c}_{\texttt{PT-LiRPA}}.$$

Proof. The proof then directly follows from Lemma 3.3 and the derivations of H. Zhang, T.-W. Weng, et al. (2018). □

3.1 A Qualitative Bound of Statistical Prediction of Tolerance Limit

The result of (Wilks 1942) allows us to bound the error of the sample-based procedure employed in our PT-LiRPA framework in terms of the number of potential violations in future samples. However, it does not provide any information on the magnitude of such possible violations with respect to the estimated bounds. Additionally, the probabilistic guarantees we provide so far are, broadly speaking, weaker than those offered by related probabilistic methods such as (Cohen et al. 2019; L. Weng, Chen, et al. 2019), where the guarantees hold for any input $\boldsymbol{x} \in C$, and not only for a predefined fraction of it. In this section, we aim to complement Wilks (1942) statistical result with a qualitative interpretation that bounds the potential error between the true minimum and its estimate based on samples. Specifically, we leverage known results on *extreme value theory* (Haan and Ferreira 2006) to strengthen and extend our original guarantee to apply to the entire perturbation set C, thus bringing the proposed solution in line with other state-of-the-art probabilistic approaches.

We start by noticing that the problem of finding an empirical minimizer close to a function's exact minimizer is well established in the statistical literature (see, e.g., (Archetti and Schoen 1984)). Exploiting quantitative assumptions on the objective function—such as a Lipschitz condition that holds in our setting—can facilitate the bounding of the possible magnitude error in the sampled-based minimizer computation approach. In this vein, we begin by providing a first worst-case qualitative bound on the maximum Δ error we can achieve when employing our sampling-based approach to compute the estimated reachable set with respect to the real (unknown) ones. For readability, in the following we simplify the notation $z_j^{(i)}$—which denotes the pre-activation value of the node in position j of layer i—by removing the indices and referring generically to a node's pre-activation value as z. We adopt the same simplification for the corresponding lower and upper bound estimates, denoted as $\bar{l}$ and $\underline{u}$, respectively.

We start with the following result.

Theorem 3.5 (Worst-case excess bound)**.** *Fix a neuron in our N-layer DNN. Let z be the function mapping the input $\boldsymbol{x}$ to the network to the pre-activation value $z(\boldsymbol{x})$ of the neuron of interest. Let $\bar{l} = \min_{k=1,\dots,n} z(\boldsymbol{x}_k)$ be the minimum pre-activation value observed in a sample of n inputs, $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ independently and uniformly drawn from $C \subseteq \mathbb{R}^d$. Let $l^* = \min_{\boldsymbol{x} \in C} z(\boldsymbol{x})$ be the actual minimum pre-activation value achievable for z over all $\boldsymbol{x} \in C$. Then, if z is Lipschitz continuous, it holds that:*

$$Pr[|\bar{l} - l^*| \leq \Delta] \geq 1 - exp\left(-n\left(\frac{\Delta}{L}\right)^d \cdot \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\right) \tag{8}$$

where L is the Lipschitz constant of the function $z()$, d is the dimension of the perturbation region C, Γ is the gamma function, and Δ the maximum error we are interested in bounding.

Proof. Since z is an L-Lipschitz function, then we have that for each $k = \{1, \dots, n\}$, and $\boldsymbol{x}^*$ being a minimizer of z, i.e., $z(\boldsymbol{x}^*) = l^*$, it holds that

$$|z(\boldsymbol{x}_k) - l^*| \leq L||\boldsymbol{x}_k - \boldsymbol{x}^*||_2.$$

From the function's Lipschitz continuity property, we have

$$\begin{aligned} Pr[|\bar{l} - l^*| \geq \Delta] &= Pr[\forall k\ z(\boldsymbol{x}_k) - z(\boldsymbol{x}^*) \geq \Delta] \\ &\leq Pr[\forall k\ L||\boldsymbol{x}_k - \boldsymbol{x}^*||_2 \geq \Delta] \\ &= Pr[\forall k\ ||\boldsymbol{x}_k - \boldsymbol{x}^*||_2 \geq \frac{\Delta}{L}] \end{aligned}$$

Fix $S = \{\boldsymbol{x} \in C : ||\boldsymbol{x} - \boldsymbol{x}^*||_2 \leq \Delta/L\}$. This is the sphere of radius Δ/L centered at $\boldsymbol{x}^*$. For a uniformly sampled point $\boldsymbol{x} \in C$, let $\mu(S) = Pr[\boldsymbol{x} \in S]$. This is equal to the volume of the set S which is proportional to $(\Delta/L)^d \cdot c(d)$, where $c(d) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$ is a constant that depends on the dimension d. Hence, by the independence of the $\boldsymbol{x}_k$s, the probability $Pr[\forall k\ \boldsymbol{x}_k \notin S] = (1 - \mu(S))^n$.

$$\begin{aligned} Pr[|\bar{l} - l^*| \geq \Delta] &\leq Pr[\forall k\ ||\boldsymbol{x}_k - \boldsymbol{x}^*||_2 \geq \frac{\Delta}{L}] \\ &= Pr[\forall k\ \boldsymbol{x}_k \notin S] \\ &= (1 - \mu(S))^n \\ &\text{from the fact that } (1-x)^n \approx e^{-nx} \\ &\leq exp\left(-n\left(\frac{\Delta}{L}\right)^d \cdot \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\right) \end{aligned}$$

Hence, $Pr[|\bar{l} - l^*| \leq \Delta] \geq 1 - Pr[|\bar{l} - l^*| \geq \Delta] \geq 1 - exp\left(-n\left(\frac{\Delta}{L}\right)^d \cdot \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\right)$. □

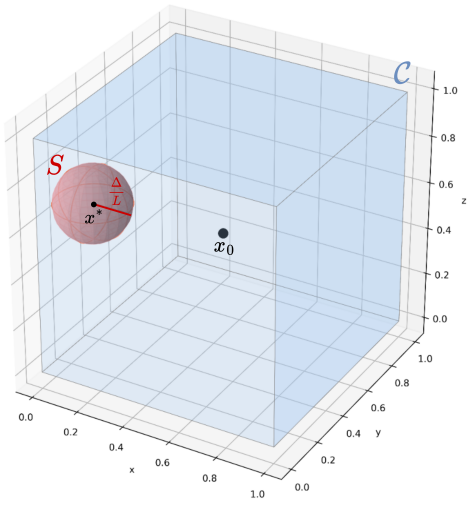

Fig. 4. Illustrative representation in 3D of Theorem 3.5.

Broadly speaking, in the theorem, we bound the probability that the estimated minimizer deviates from the true minimizer by more than a predefined threshold Δ. Owing to the Lipschitz properties of DNNs, this is equivalent to estimating the probability that a random sample of inputs includes at least one point within distance Δ/L of the true minimizer $\mathbf{x}^*$, i.e., an input $\boldsymbol{x}$ in $S = \{\mathbf{x} \in C : \|\mathbf{x} - \mathbf{x}^*\|_2 \leq \Delta/L\}$. Geometrically, S corresponds to the intersection of the perturbation region C with a ball of radius Δ/L centered at $\mathbf{x}^*$. If $\mathbf{x}^*$ lies sufficiently far from the boundary of C, then S is exactly such a ball. Otherwise, if $\mathbf{x}^*$ is near the boundary, S becomes a truncated ball, i.e., the intersection with C. This is clearly represented in Fig. 4, where we depict a potential perturbation region C in 3d derived from the original input $\boldsymbol{x}_0$ and the minimizer $\boldsymbol{x}^*$ of f for C.

Clearly, the bound given in Theorem 3.5 accounts for the worst-case scenario and assumes no specific property of the distribution on the input. When the perturbation region has high dimensionality or the DNN has a large Lipschitz constant, this bound may become too loose to offer meaningful insights. Following EVT, and particularly the results of (De Haan 1981), we can provide a tighter bound on the error in the estimation.

Lemma 3.6 (Tighter Qualitative bound)**.** *Fix a neuron in our N-layer DNN and let z be the be the function mapping the input network to the pre-activation value of the neuron. Let $\bar{l} = \min_{k=1,\ldots,n} z(\boldsymbol{x}_k)$ the minimum pre-activation value observed in a sample of n inputs, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ independently and uniformly drawn from $C \subseteq \mathbb{R}^d$. Let $l^* = \min_{\boldsymbol{x} \in C} z(\boldsymbol{x})$, the true minimum pre-activation value achievable over all $\boldsymbol{x} \in C$. Let $Y_1, Y_2, \ldots, Y_n$ be the order statistics for $z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_n)$. For all $p \in (0, 1)$, then it holds that:*

$$Pr\left[|\bar{l} - l^*| \leq \frac{Y_2 - Y_1}{(1-p)^{-a} - 1}\right] \geq 1 - p \tag{9}$$

with $a \approx log(v)/log\left(\frac{Y_v - Y_3}{Y_3 - Y_2}\right)$, with v any integer valued function of n such that $v(n) \to \infty$ and $v(n)/n \to 0$.

Importantly, Lemma 3.6, holds under the assumption that the samples $Y_1, \ldots, Y_n$ follow a nondegenerate limit distribution function in the form $1 - exp(-x^a)$ for some $a > 0$ (De Haan 1981). In particular, this is true for uniformly differentiable f, which is the case for most common neural networks, e.g., when the activation functions are Sigmoid, Tanh, etc. For ReLU-based networks, where the differentiability requirement is not satisfied, we can still apply Lemma 3.6 by splitting the non-linearities, i.e., splitting all the ReLU nodes, achieving a linear system differentiable everywhere (Bunel et al. 2020).

Concretely, whenever a ReLU's interval bound crosses 0 (i.e., $\underline{z} < 0 < \overline{z}$) computed over the perturbation region C, we split the computation into two linear branches enforcing $z(\boldsymbol{x}) \geq 0$ (active) and $z(\boldsymbol{x}) \leq 0$ (inactive). On each resulting subregion, the network is affine, so the neuron's pre-activation is an affine (hence differentiable) map of $\boldsymbol{x}$. Sampling i.i.d. uniformly from C and rejecting points that do not satisfy a branch's linear constraints is equivalent to sampling i.i.d. uniformly from that constrained subregion. Hence, the order statistics restricted to the subregion obey the required extreme-value limit. In practice, in our experimental setting described in Sec. 5, where C is an ℓ_∞-ball and ReLU constraint splitting is applied, the rejection rate remains very low. Specifically, the fraction of discarded samples is always negligible (typically below 5% on average). This is because the probability that a uniformly sampled point violates the constraints decreases rapidly as the sample size increases, and the rejection step only eliminates points outside the feasible region while preserving the independence of the remaining samples.

3.2 Extending Wilks' Probabilistic Guarantees

Building on the results of Lemma 3.6, we can extend the theoretical guarantees of `PT-LiRPA` to the entire perturbation region C. Specifically, Lemma 3.6 provides a bound on the maximum error in estimating the true minimum (or maximum) of a given intermediate node z. By adding this EVT-based error bound to the initial estimates $\bar{l}$ and $\underline{u}$, obtained from the first random sampling of n inputs from C, we derive a probabilistically tight

approximation of the reachable set for z that holds for any $\boldsymbol{x} \in C$. Notably, following the asymptotic requirements in (Haan and Ferreira 2006), we can choose the number of upper order statistics as $\nu(n) = \lfloor n^{\xi} \rfloor$, with $\xi \in (0,1)$, which clearly satisfies the conditions $\nu(n) \to \infty$ and $\nu(n)/n \to 0$.

Hence, for any intermediate neuron in the network, we have:

Theorem 3.7 (Improved PT-LiRPA probabilistic guarantee on the estimated reachable set)**.** *Fix a positive integer n and real values $p, \xi \in (0,1)$, and let $\nu = \lfloor n^{\xi} \rfloor$. For any neuron z in an N-layer DNN, let $z(\boldsymbol{x})$ denote the pre-activation value of z when $\boldsymbol{x}$ is the input to the network. Let $\bar{l} = \min_{k=1,\dots,n} z(\boldsymbol{x}_k)$ and $\underline{u} = \max_{k=1,\dots,n} z(\boldsymbol{x}_k)$ as the minimum and maximum pre-activation values observed in a sample of n random inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ independently and uniformly drawn from $C \subseteq \mathbb{R}^d$. Let $Y_1 \le Y_2 \le \cdots \le Y_n$ be the order statistics for the observed values $z(\boldsymbol{x}_k)$.*

Then for any $\boldsymbol{x} \in C$ it holds that:

$$Pr\left[\bar{l} - \frac{Y_2 - Y_1}{(1-p)^{-a_l} - 1} \le z(\boldsymbol{x}) \le \underline{u} + \frac{Y_n - Y_{n-1}}{(1-p)^{-a_u} - 1}\right] \ge 1 - 2p.$$

with $a_l \approx \frac{\log(\nu)}{\log\left(\frac{Y_\nu - Y_3}{Y_3 - Y_2}\right)}$, $a_u \approx \frac{\log(\nu)}{\log\left(\frac{Y_{n-2} - Y_{n-\nu}}{Y_{n-1} - Y_{n-2}}\right)}$.

Proof. The proof directly follows from the union bound, exploiting Lemma 3.6. □

Therefore, in a network with m neurons, using the LiRPA approach employing the estimates $\hat{l}$ and $\hat{u}$ for the reachable set of neuron z as given by Theorem 3.7, i.e.,

$$\hat{l} = \bar{l} - \frac{Y_2 - Y_1}{(1-p)^{-a_l} - 1} \qquad \hat{u} = \underline{u} + \frac{Y_n - Y_{n-1}}{(1-p)^{-a_u} - 1}, \tag{10}$$

we can compute a linear lower bound function $\boldsymbol{a}^T_{\texttt{PT-LiRPA}}(\boldsymbol{x}) + \boldsymbol{c}_{\texttt{PT-LiRPA}}$ such that with probability at least $1 - 2mp$ for any $\boldsymbol{x} \in C$ satisfies:

$$f(\boldsymbol{x}) \ge \boldsymbol{a}^T_{\texttt{PT-LiRPA}}(\boldsymbol{x}) + \boldsymbol{c}_{\texttt{PT-LiRPA}}. \tag{11}$$

We apply the sampling procedure independently for each of the m neurons, estimating the reachable set using n i.i.d. samples drawn from C. By Theorem 3.7, each reachable set is correct with probability at least $1 - 2p$. Then we use union bound to estimate the probability that all reachable sets are simultaneously correct, yielding a lower bound $1 - 2mp$. We note that this analysis does not need independence between estimation errors across neurons. In fact, dependencies may arise due to the layered nature of DNNs, which may lead to compounding errors in deeper layers.

Some observations are in order. The result of Theorem 3.7 provides for any choice of n and p a guarantee holding for all input values $x \in C$. However, there is a potential weakness in its practical use since we would also like to guarantee that the two sides of inequality (11) are as close as possible, i.e., that the linear lower bound is as tight as possible. For this, we would like to have that for each neuron z the values $\hat{l}$ (respectively, $\hat{u}$) and $\bar{l}$ (respectively, $\underline{u}$) are close. However, the tail index parameters a_l and a_u, ruling their difference, depend on the shape of the tail of the distribution of $z(\boldsymbol{x})$ over C. This precludes the possibility of computing the minimum value of n yielding to achieve a desired precision for a given desired confidence p.

One possibility to address this issue is to guess the minimum number of samples by a standard doubling technique: keep on doubling the number of samples used until the estimated tail corrections fall below a desired threshold. Alternatively, we can start with a conservative sample size n inspired by Wilks' formula and our extension (Proposition 3.2), set $\nu = \lfloor n^{\xi} \rfloor$ with $\xi \in (0,1)$, and compute the resulting values $\hat{l}$, $\hat{u}$ as by Theorem 3.7. While potentially suboptimal, the experiments show that this approach produces linear bounds that, besides satisfying the above guarantees, remain significantly tighter than those provided by the traditional LiRPA-based approach, especially in deeper layers where LiRPA errors tend to compound.

3.3 Example of PT-LiRPA Linear Bounds Computation.

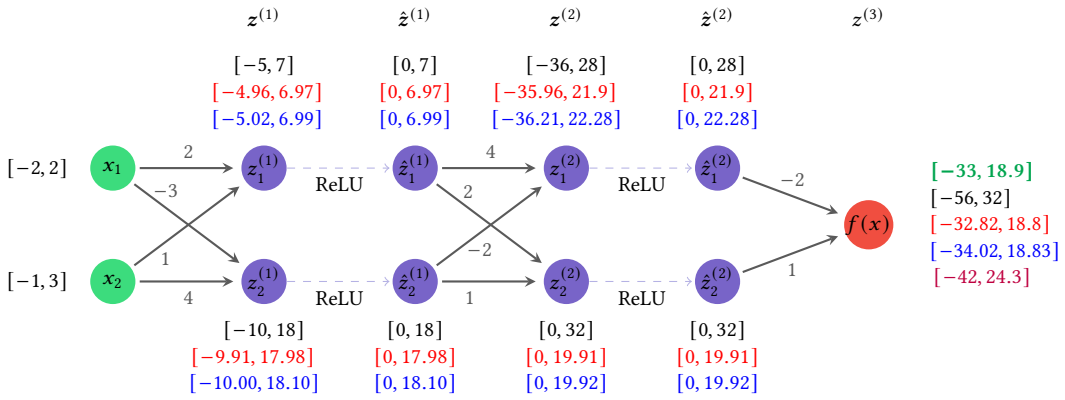

Fig. 5. Toy DNN used in this example. Intervals reported in green are the exact output reachable set computed via MIP, in black are the results of the IBP, and in purple the CROWN ones considering the input $[[-2, 2], [-1, 3]]$. In red are the reachable sets computed using a naive sampling-based approach of $n = 10k$ samples. Finally, in blue, the ones computed using a naive sampling-based approach combined with the EVT error estimation.

Recalling the example provided in Sec. 2.3, we show now the computation of the linear bounds employing CROWN enhanced with PT-LiRPA. In detail, the calculation is analogous to what we have seen above, except for the construction of the diagonal matrices and bias vectors. In the following, we will compute the linear bounds using both the theoretical results of Theorem 3.4 and Theorem 3.7.

We start by computing the estimated reachable sets from a sample-based approach in C using $n = 10k$ samples, which for the Proposition 3.2, with $R = 0.999$ and considering the number of neurons in the DNN, are sufficient to have a final confidence $\psi \geq 0.99$. We report in Figure 5 highlighted in red the estimated reachable sets obtained from the propagation of n random samples drawn from $[[-2, 2], [-1, 3]]$. As we can notice, the bounds are slightly tighter than the overestimated ones obtained from the IBP process. Our intuition is thus that from the computation of $\underline{\boldsymbol{D}}^{(i)}, \overline{\boldsymbol{D}}^{(i)}, \underline{\boldsymbol{b}}^{(i)}, \overline{\boldsymbol{b}}^{(i)}$ using these tightened bounds we can obtain more accurate lower and upper final linear bounds. For the diagonal matrices and bias vectors, we get:

$$\underline{\boldsymbol{D}}^{(2)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3784 & 0 \\ 0 & 1 \end{bmatrix} \qquad \underline{\boldsymbol{b}}^{(2)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 13.61 \\ 0 \end{bmatrix}$$

$$\overline{\boldsymbol{D}}^{(2)} = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \overline{\boldsymbol{b}}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and

$$\underline{\boldsymbol{D}}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \alpha \end{bmatrix} = \begin{bmatrix} 0.5839 & 0 \\ 0 & 0 \end{bmatrix} \qquad \underline{\boldsymbol{b}}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 2.8986 \\ 0 \end{bmatrix}$$

$$\overline{\boldsymbol{D}}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \frac{u}{u-l} \end{bmatrix} = \begin{bmatrix} 0.5839 & 0 \\ 0 & 0.6448 \end{bmatrix} \qquad \overline{\boldsymbol{b}}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ \frac{-ul}{u-l} \end{bmatrix} = \begin{bmatrix} 2.8986 \\ 6.3870 \end{bmatrix}.$$

Thus computing all the $\boldsymbol{A}$s and ds vectors,

$$\underline{\boldsymbol{A}}^{(2)} = \underline{\boldsymbol{A}}^{(3)}\underline{\boldsymbol{D}}^{(2)}\boldsymbol{W}^{(2)} = [-1.0275, 2.5138], \qquad \overline{\boldsymbol{A}}^{(2)} = \overline{\boldsymbol{A}}^{(3)}\overline{\boldsymbol{D}}^{(2)}\boldsymbol{W}^{(2)} = [2, 1],$$
$$\underline{\boldsymbol{A}}^{(1)} = \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{D}}^{(1)}\boldsymbol{W}^{(1)} = [-1.2, -0.6], \qquad \overline{\boldsymbol{A}}^{(1)} = \overline{\boldsymbol{A}}^{(2)}\overline{\boldsymbol{D}}^{(1)}\boldsymbol{W}^{(1)} = [0.4013, 3.7471],$$
$$\underline{d} = \underline{\boldsymbol{A}}^{(3)}\underline{\boldsymbol{b}}^{(2)} + \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{b}}^{(1)} = -30.1983, \qquad \overline{d} = \overline{\boldsymbol{A}}^{(3)}\overline{\boldsymbol{b}}^{(2)} + \overline{\boldsymbol{A}}^{(2)}\overline{\boldsymbol{b}}^{(1)} = 12.1841,$$

we obtain:

$$\underline{f}_{\text{CROWN w/ PT-LiRPA}} = \min_{\boldsymbol{x}\in C} \underline{\boldsymbol{A}}^{(1)}(\boldsymbol{x}) + \underline{d} = -||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \underline{\boldsymbol{A}}^{(1)}\boldsymbol{x}_0 + \underline{d}$$
$$= -3.6 - 0.6 - 30.1983 = -34.4.$$

$$\overline{f}_{\text{CROWN w/ PT-LiRPA}} = \max_{\boldsymbol{x}\in C} \overline{\boldsymbol{A}}^{(1)}(\boldsymbol{x}) + \overline{d} = ||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \overline{\boldsymbol{A}}^{(1)}\boldsymbol{x}_0 + \overline{d}$$
$$= 8.2968 + 3.7471 + 12.1841 = 24.23.$$

As we can notice, these bounds are significantly tighter than the ones computed using CROWN ($[-42, 24.3]$). However, the theoretical guarantees provided by Theorem 3.4 allow us to state that these bounds are probabilistically sound only for a fraction R of the perturbation region C, thus resulting in a slightly weaker guarantee w.r.t. the one provided by existing probabilistic approaches. Nonetheless, we believe that if one accepts the assumption underlying this theoretical guarantee, this approach still presents a valuable and computationally efficient tool for computing probabilistically valid linear output bounds. In the following, we show how to practically extend these theoretical guarantees to the whole perturbation region, exploiting Theorem 3.7.

We start again from computing the estimated reachable sets from a sample-based approach in C. For each estimated lower and upper bound, we compute and add the corresponding error using $\frac{Y_2 - Y_1}{(1-p)^{-a_l} - 1}$ for the lower and $\frac{Y_n - Y_{n-1}}{(1-p)^{-a_u} - 1}$ for the upper bound, respectively. We report in Figure 5 highlighted in blue the new estimated reachable sets obtained from the propagation of n random samples drawn from $[[-2, 2], [-1, 3]]$ with the addition of the corresponding error. Hence, we speculate that recomputing $\underline{\boldsymbol{D}}^{(i)}, \overline{\boldsymbol{D}}^{(i)}, \underline{\boldsymbol{b}}^{(i)}, \overline{\boldsymbol{b}}^{(i)}$ using these new estimated reachable sets we can still obtain more accurate lower and upper final linear bounds w.r.t. the LiRPA-based approaches. In fact, we obtain:

$$\underline{\boldsymbol{D}}^{(2)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3809 & 0 \\ 0 & 1 \end{bmatrix} \qquad \underline{\boldsymbol{b}}^{(2)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 13.7919 \\ 0 \end{bmatrix}$$
$$\overline{\boldsymbol{D}}^{(2)} = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad \overline{\boldsymbol{b}}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and

$$\underline{\boldsymbol{D}}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \alpha \end{bmatrix} = \begin{bmatrix} 0.5820 & 0 \\ 0 & 0 \end{bmatrix} \qquad \underline{\boldsymbol{b}}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ 0 \end{bmatrix} = \begin{bmatrix} 2.9219 \\ 0 \end{bmatrix}$$
$$\overline{\boldsymbol{D}}^{(1)} = \begin{bmatrix} \frac{u}{u-l} & 0 \\ 0 & \frac{u}{u-l} \end{bmatrix} = \begin{bmatrix} 0.5820 & 0 \\ 0 & 0.6441 \end{bmatrix} \qquad \overline{\boldsymbol{b}}^{(1)} = \begin{bmatrix} \frac{-ul}{u-l} \\ \frac{-ul}{u-l} \end{bmatrix} = \begin{bmatrix} 2.9219 \\ 6.4427 \end{bmatrix}.$$

We can now compute all the $\boldsymbol{A}$s and ds vectors.

$$\begin{aligned}
\underline{\boldsymbol{A}}^{(2)} &= \underline{\boldsymbol{A}}^{(3)}\underline{\boldsymbol{D}}^{(2)}\boldsymbol{W}^{(2)} = [-1.0471, 2.5235] \\
\overline{\boldsymbol{A}}^{(2)} &= \overline{\boldsymbol{A}}^{(3)}\overline{\boldsymbol{D}}^{(2)}\boldsymbol{W}^{(2)} = [2, 1] \\
\underline{\boldsymbol{A}}^{(1)} &= \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{D}}^{(1)}\boldsymbol{W}^{(1)} = [-1.2187, -0.6094] \\
\overline{\boldsymbol{A}}^{(1)} &= \overline{\boldsymbol{A}}^{(2)}\overline{\boldsymbol{D}}^{(1)}\boldsymbol{W}^{(1)} = [0.3957, 3.7402] \\
\underline{d} &= \underline{\boldsymbol{A}}^{(3)}\underline{\boldsymbol{b}}^{(2)} + \underline{\boldsymbol{A}}^{(2)}\underline{\boldsymbol{b}}^{(1)} = -30.6434 \\
\overline{d} &= \overline{\boldsymbol{A}}^{(3)}\overline{\boldsymbol{b}}^{(2)} + \overline{\boldsymbol{A}}^{(2)}\overline{\boldsymbol{b}}^{(1)} = 12.2865
\end{aligned}$$

Finally we have

$$\begin{aligned}
\underline{f}_{\text{PT-LiRPA}} = \min_{\boldsymbol{x}\in C} \underline{\boldsymbol{A}}^{(1)}(\boldsymbol{x}) + \underline{d} &= -||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \underline{\boldsymbol{A}}^{(1)}\boldsymbol{x}_0 + \underline{d} \\
&= -3.6562 - 0.6094 - 30.6434 = -34.91.
\end{aligned}$$

$$\begin{aligned}
\overline{f}_{\text{PT-LiRPA}} = \max_{\boldsymbol{x}\in C} \overline{\boldsymbol{A}}^{(1)}(\boldsymbol{x}) + \overline{d} &= ||\underline{\boldsymbol{A}}^{(1)}||_1 \cdot \varepsilon + \overline{\boldsymbol{A}}^{(1)}\boldsymbol{x}_0 + \overline{d} \\
&= 8.2717 + 3.7402 + 12.2865 = 24.3.
\end{aligned}$$

Although the upper bound is equivalent to the original CROWN approach, we can notice that our procedure produces a tighter lower bound. This toy example provides a preliminary insight into the potential of the proposed solution. Our speculation on the impact of PT-LiRPA on realistic verification instances will be confirmed by the experiments presented in Sec. 5.

EVT-based approach to directly bound the output? A natural question that arises is whether the results of Theorem 3.7 can be used directly to obtain a tight estimation of the output reachable set, without relying on the LiRPA combination. Although this sampling-based method offers a probabilistic estimate that, with high confidence, contains the entire perturbation region, it may still underestimate the true output bounds due to its reliance on a finite number of samples. For example, the MIP result yields bounds of $[-33.0, 18.9]$, and as we can notice in Fig. 5 highlighted in blues, the output reachable set only applying a forward computation of Theorem 3.7 still underestimates the exact upper bound $[-34.02, 18.83]$. In contrast, since our method integrates a sampling-based approach with any LiRPA method—which inherently provides sound overestimations—the final computed bounds will always be at least as tight as those obtained through estimation based on a finite number of samples and are likely (with a confidence at least $1 - 2mp$) to produce valid linear bounds, i.e., not discarded by potential adversarial attacks (in fact, we obtain as final result $[-34.4, 24.23]$). Additionally, as emphasized in prior work (Xu, Shi, et al. 2020), combining forward and backward analysis typically yields tighter bounds compared to using a simple forward bound computation. This observation further motivated our investigation into how tighter reachable sets can enhance the linearization approaches for verification efficiency.

4 PT-LiRPA Framework for Neural Network Verification

Based on the theoretical results of Sec. 3, we now present in Algorithm 3 the PT-LiRPA approach for the verification process. For the sake of clarity and without loss of generality, we present the procedure applied to the parallel BaB as shown for the optimized LiRPA approach proposed in (Xu, H. Zhang, et al. 2021).

Algorithm 2 PGD_Attack(Madry et al. 2018)

Input Original input $\boldsymbol{x}_0$, neural network f, maximum perturbation ε to create data range $[x_{\min}, x_{\max}]$, step size α, iterations T, *random_start*
Output adversarial example $\boldsymbol{x}_{\text{adv}}$ with $\|\boldsymbol{x}_{\text{adv}} - \boldsymbol{x}_0\|_\infty \leq \varepsilon$ or original input $\boldsymbol{x}_0$
if *random_start* **then**
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0 + \text{Uniform}(-\varepsilon, \varepsilon)$
else
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
$\boldsymbol{x}_{\text{adv}} \leftarrow \text{clip}(\boldsymbol{x}, x_{\min}, x_{\max})$
for $t \in \{1, \ldots, T\}$ **do**
  if $f(\boldsymbol{x}_{\text{adv}}) \leq 0$ **then**
    return $\boldsymbol{x}_{\text{adv}}$
  $g \leftarrow \nabla_{\boldsymbol{x}_{\text{adv}}} f(\boldsymbol{x}_{\text{adv}})$ ▷ gradient of scalar output w.r.t. adversarial input
  $\boldsymbol{x}_{\text{adv}} \leftarrow \boldsymbol{x}_{\text{adv}} - \alpha \cdot \text{sign}(g)$ ▷ descent step for minimizing s
  $\Delta \leftarrow \text{clip}(\boldsymbol{x}_{\text{adv}} - \boldsymbol{x}_0, -\varepsilon, \varepsilon)$ ▷ project perturbation onto L_∞ ball
  $\boldsymbol{x}_{\text{adv}} \leftarrow \text{clip}(\boldsymbol{x}_0 + \Delta, x_{\min}, x_{\max})$
return $\boldsymbol{x}_0$

Given a DNN f with m neurons and a region of interest C, the verification process typically involves a projected gradient descent (PGD) attack (Madry et al. 2018). This attack, reported in Alg. 2 for the sake of completeness, can be employed before, after, or during the BaB procedure to search for potential adversarial inputs within the input region under consideration.

In detail, we report in our PT-LiRPA algorithm (Alg. 3), a potential employment of PGD during the verification process. Specifically, the attack is performed before and during the BaB, performing a projected gradient descent

Algorithm 3 PT-LiRPA on parallel BaB

Input: A DNN f with N layers and m neurons, an original input $\boldsymbol{x}_0$, a maximum ε perturbation to create a perturbation region C, maximum error in the confidence p, sample size n, $\xi \in (0, 1)$ for $\nu(n)$ and a batch size t.
Output: *robust*/*not-robust*
▷ as in Alg. 2
if PGD_attack(f, C) **then**
  return *not robust*
interm_bounds $\leftarrow$ get_interm_bounds(f, C, n, p, ξ) ▷ as in Alg. 4
$\underline{f}_C, \overline{f}_C \leftarrow$ LiRPA$(f, C,$ *interm_bounds*$)$ ▷ as in Alg. 1 where C contains $\boldsymbol{x}_0, \varepsilon$
$\mathcal{B} \leftarrow (\underline{f}_C, \overline{f}_C)$
while $\mathcal{B} \neq \emptyset$ **do**
  $C_1, \ldots, C_t \leftarrow$ split$(\mathcal{B}, t)$
  $interm\_bounds_{C_1,\ldots,C_t} \leftarrow$ get_interm_bounds$(f, [C_1, \ldots, C_t], n, p, \xi)$
  $(\underline{f}_{C_1}, \overline{f}_{C_1}), \ldots, (\underline{f}_{C_t}, \overline{f}_{C_t}) \leftarrow$ LiRPA$(f, [C_1, \ldots, C_t], interm_bounds_{C_1,\ldots,C_t})$ ▷ parallel exec. of Alg. 1 on $C_{1,\ldots,t}$
  $\mathcal{B}' \leftarrow (\underline{f}_{C_1}, \overline{f}_{C_1}), \ldots, (\underline{f}_{C_t}, \overline{f}_{C_t})$
  if $\exists\, C_i \in \mathcal{B}'$ *s.t.* $\overline{f}_{C_i} < 0$ or PGD_attack(f, C_i) **then**
    return *not robust*
  $\mathcal{B} \leftarrow \mathcal{B}' \setminus$ get_robust_domains$(\mathcal{B}')$
return *robust*

search in the L_∞ ball $C = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \varepsilon\}$ to find an adversarial input $\boldsymbol{x}_{\text{adv}}$ that makes the scalar model output non-positive, i.e., $f(\boldsymbol{x}_{\text{adv}}) \leq 0$. Starting from either the clean original input or a random uniform perturbation in $[-\varepsilon, \varepsilon]$ (i.e., a random input vector in the C) the method iteratively evaluates the scalar output, computes its gradient with respect to the input, and takes an L_∞-constrained descent step using the elementwise sign of the gradient. After each update, the perturbation is projected back onto the L_∞ ball and the input is clipped to the valid data range; the procedure stops early if a negative output is obtained and otherwise runs for at most T iterations.
The main hyperparameters are the maximum perturbation ε, the step size α (typically chosen on the order of ε/T), the maximum iterations T, and the optional random start; multiple restarts or momentum can be used to increase attack strength. Success provides a concrete counterexample to robustness within the prescribed L_∞ radius, while failure is only a heuristic indication and does not constitute a formal certificate of robustness.

Hence, Alg. 3 begins with a PGD attack (lines 3-5), and if no adversarial is found, we proceed with the Branch-and-Bound process. We compute the estimated reachable sets using the `get_interm_bounds` method (line 6), which exploits the results of Theorem 3.7 and is reported here below in Alg. 4 for clarity.

Algorithm 4 `get_interm_bounds`

interm_bounds $\leftarrow \{\}$
$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \leftarrow \text{UniformSampling}(C, n)$ ▷ collect n random i.i.d inputs from C
for each intermediate layer **do**
  $\hat{\boldsymbol{l}}, \hat{\boldsymbol{u}} \leftarrow \{\}$
  for each node z in layer nodes **do**
    $Y_1, \ldots, Y_n \leftarrow \text{Sort}(z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_n))$ ▷ with $z(\cdot)$ as in Theorem 3.7
    $\overline{l}, \underline{u} \leftarrow Y_1, Y_n$
    $a_l, a_u \leftarrow \frac{\log(\nu)}{\log\left(\frac{Y_\nu - Y_3}{Y_3 - Y_2}\right)}, \frac{\log(\nu)}{\log\left(\frac{Y_{n-2} - Y_{n-\nu}}{Y_{n-1} - Y_{n-2}}\right)}$
    $\hat{l}, \hat{u} \leftarrow \overline{l} - \frac{Y_2 - Y_1}{(1-p)^{-a_l} - 1}, \underline{u} + \frac{Y_n - Y_{n-1}}{(1-p)^{-a_u} - 1}$ ▷ as in Eq. 10
    $\hat{\boldsymbol{l}}, \hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{l}} \cup \hat{l}, \hat{\boldsymbol{u}} \cup \hat{u}$
  interm_bounds $\leftarrow$ *interm_bounds* $\cup [\hat{\boldsymbol{l}}, \hat{\boldsymbol{u}}]$ ▷ store the vector of lower and upper bounds for the specific layer
return *interm_bounds*

We then use these bounds in the linear bounds computation on any existing `LiRPA` approach (line 6), following Alg. 1 of Sec. 2.3 and the computation shown in the toy example of Sec. 3.3. We store the resulting output bounds $\underline{f}$ and $\overline{f}$ for the region C, namely $\underline{f}_C$ and $\overline{f}_C$ in a set $\mathcal{B}$ of unverified regions (line 7). We then continue the BaB process by splitting (using the `split` method) the original region from $\mathcal{B}$ into t sub-regions (line 9). Notably, we can perform the parallel selection and splitting into sub-domains using information on unstable ReLU nodes, as shown in (Bunel et al. 2020; S. Wang, H. Zhang, et al. 2021), or just on the perturbation region C_i (S. Wang, Pei, et al. 2018). Once we have the new sub-domains, we recompute the estimated reachable sets in parallel and use these bounds for the new computation of the linear lower and upper bounds for each sub-region, and

Algorithm 5 `get_robust_domains`

robust_domains $\leftarrow \{\}$
for $(\underline{f}_{C_i}, \overline{f}_{C_i}) \in \mathcal{B}$ **do**
  if $\underline{f}_{C_i} > 0$ **then**
    robust_domains $\leftarrow$ *robust_domains* $\cup\, C_i$ ▷ following Def. 2.1
return *robust_domains*

we update $\mathcal{B}$ with the resulting unverified sub-domains (lines 10-12). At each iteration, the process can end either because there is at least a single sub-domain $C_i \in \mathcal{B}$ that presents $\overline{f} < 0$, or a PGD attack succeeds, thus returning *not robust* as the answer (lines 13-16). Otherwise, the process continues updating $\mathcal{B}$ with the unverified domains using the procedure `get_robust_domains` (line 15) reported in Alg. 5. Following Theorem 3.7, if we reach the emptiness of $\mathcal{B}$, thus no adversarial examples are found during the verification process and all the sub-domains are evaluated as *robust*, we can state that, with a confidence $\geq 1 - 2mp$, the DNN is robust for the whole perturbation region C.

5 Empirical Evaluation

Our empirical evaluation consists of three main experiments to answer the following questions:

- **Q1.** *How does the hyperparameter ξ impact the lower bound computation? How does the number of samples employed in the computation process impact the tightening process?*
- **Q2.** *How much* `PT-LiRPA` *improves the robustness bounds certificates w.r.t. other probabilistic and worst-case methods? What is the computational overhead of the proposed solution with respect to a worst-case certification approach?*
- **Q3.** *What is the general impact of* `PT-LiRPA` *in the verification process of challenging instances such as the one employed in the VNN-COMP (Brix et al. 2023; Müller et al. 2022)?*

All data are collected on a cluster running Rocky Linux 9.34 equipped with Nvidia RTX A6000 (48 GiB) and a CPU AMD Epyc 7313 (16 cores). The code, trained models, and comprehensive instructions for reproducing our results is available at https://github.com/lmarza/ProbVerNet.

Answers to Q1. To address the first question, we consider the original pre-trained models on MNIST dataset. Specifically, we focus on MLP models with varying depths and activation functions. For consistency and ease of comparison, we adopt the same notation as (H. Zhang, T.-W. Weng, et al. 2018) and (L. Weng, Chen, et al. 2019): a model is denoted by the dataset name, followed by the number of layers i, the number of neurons per layer j, formatted as $i \times [j]$, and the activation function used. We begin by analyzing the mean percentage error in estimating the lower bounds of the intermediate reachable sets (using Eq. 12) for a fixed input image, using `PT-LiRPA` on various neural networks trained on the MNIST dataset.

$$error = \frac{Y_2 - Y_1}{(1-p)^{-a} - 1} \tag{12}$$

with $a \approx \frac{\log(v)}{\log\left(\frac{Y_v - Y_3}{Y_3 - Y_2}\right)}$, where $v = \lfloor n^{\xi} \rfloor$, $\xi \in (0, 1)$.

In particular, we first study the impact of the $\xi \in (0, 1)$ hyperparameter, which controls the number of order statistics $v = \lfloor n^{\xi} \rfloor$ used to estimate the tail distribution and compute the mean distance error across all intermediate nodes, and thus the neural network. Our results reported in Fig. 6 show that, using a fixed sample size of $n = 10k$, for small values of ξ, the number of extreme samples is limited, leading to high variance and bias in the tail modeling, and consequently to considerable estimation errors. As ξ increases in the interval $(0.2, 0.6)$, the error decreases significantly due to the improved reliability of the extreme value statistics. For larger ξ values, around $[0.6, 1)$, the error stabilizes below the overall mean error across all different ξ values, indicating that the estimator becomes robust and further improvements are marginal. We highlight that, even without access to the true lower (or upper, respectively) bound, one can in principle fine-tune the ξ parameter as desired to minimize the error with respect to the target value for a given application. Based on these observations, we set $\xi = 0.85$ for the following experiments, as it provides a good trade-off between low error estimation and stability across different models.

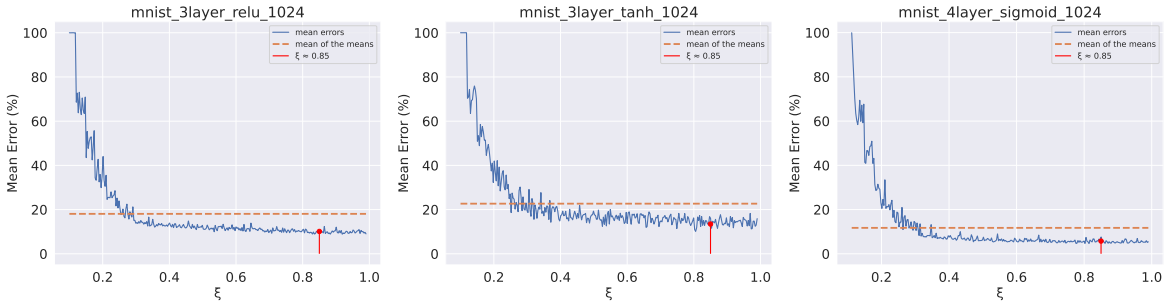

Fig. 6. Mean distance error (%) achieved on each intermediate node using PT-LiRPA with EVT-based error computation, for networks with ReLU (left), Tanh (middle), and Sigmoid (right) activations. In red we report the value $\xi = 0.85$ selected for the following experiments.

Hence, for the fixed value ξ and perturbed input, we investigate the impact of three sample sizes, namely 10000, 100000, and 350000, in the lower bound estimation of each estimated reachable set. For each sample size, we compute the mean distance error across the entire network setting $p = 0.01$, thus ensuring a confidence level of at least 99% in the final results. For the distance error achieved on each intermediate node of the network using Eq. 12 with $\xi = 0.85$. We then compute the percentage error relative to the smallest observed value in the sample. Specifically, for each neuron, the error is normalized by the absolute value of the smallest pre-activation observed value.

Table 1. Mean maximum error in estimating the lower bound of the intermediate reachable set for a fixed input image, using PT-LiRPA with $1 - p = 0.99$ on various neural networks trained on the MNIST dataset with different sample sizes.

Name model	**# samples**	**mean distance error (%)**	**# samples**	**mean distance error (%)**	**# samples**	**mean distance error (%)**
MNIST 2×[1024], ReLU	10000	12.91	100000	9.35	350000	8.78
MNIST 3×[1024], ReLU	10000	9.63	100000	8.49	350000	7.09
MNIST 4×[1024], ReLU	10000	11.89	100000	9.34	350000	8.85
MNIST 2×[1024], Tanh	10000	11.31	100000	12.22	350000	10.49
MNIST 3×[1024], Tanh	10000	10.67	100000	11.01	350000	10.34
MNIST 4×[1024], Sigmoid	10000	5.03	100000	6.21	350000	4.36
	mean error	10.24%	**mean error**	9.44%	**mean error**	8.32%

The results of Tab. 1 demonstrate that with high confidence (i.e., at least 99%) across all tested networks, increasing the sample size, the percentage error in the estimation decreases. Importantly, even with a limited sample size (e.g., 10000 samples), we note that the mean error across all the networks in the lower bound estimation is 10.24% from the estimated one, while increasing the sample size, we reach a maximum error of 8.32%. To assess the practical impact of the estimation error on the final output bounds, we consider the same perturbed input of the previous experiments and fix the sample size to $350k$. We then compare the final output bounds obtained by an exact MIP-based verification (Tjeng et al. 2017), the state-of-the-art α-CROWN (Xu, H. Zhang, et al. 2021) method, and α-CROWN (Xu, H. Zhang, et al. 2021) enhanced with our PT-LiRPA. As a representative case, we select the *MNIST_2x[1024]_ReLU* network. The reason for selecting this network is to enable a comparison with MIP (Tjeng et al. 2017), which provides exact output bounds. Since MIP solvers are inherently designed for ReLU, and more generally, piecewise-linear activations, they are not directly applicable to networks with Sigmoid or Tanh activations. Among the models listed in Table 1, the ones with larger estimation errors under ReLU

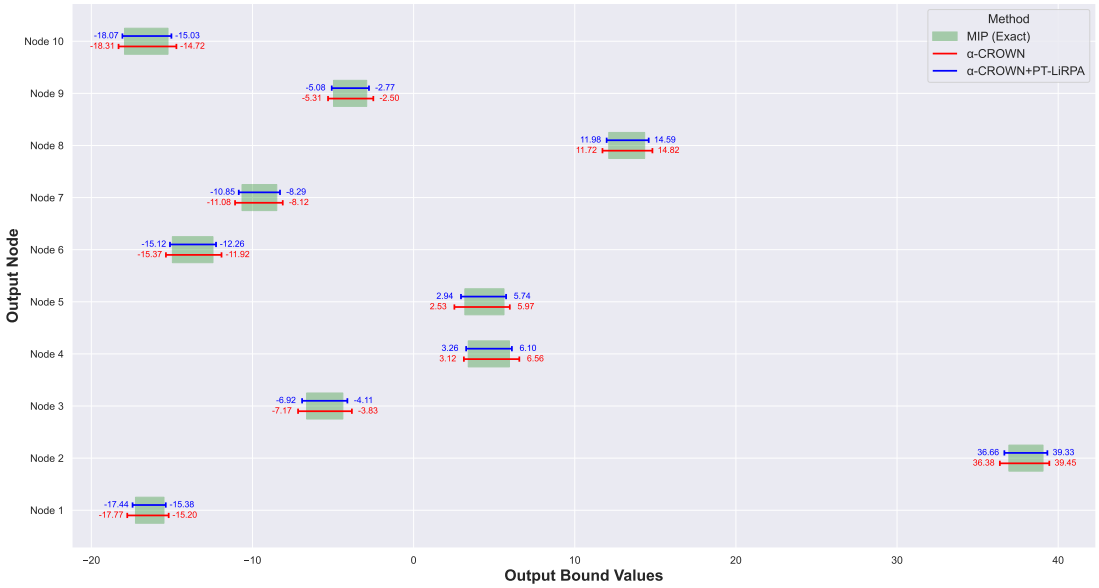

Fig. 7. Comparison of output bounds on the *MNIST_2x[1024]_ReLU* network using α-CROWN (Xu, H. Zhang, et al. 2021) reported in red,α-CROWN (Xu, H. Zhang, et al. 2021) with PT-LiRPA in blue, and exact MIP verification (Tjeng et al. 2017) in green.

activations with confidence $1 - p = 0.99$ are *MNIST_2x[1024]_ReLU* and *MNIST_4x[1024]_ReLU*. We select the former because MIP scalability becomes a limiting factor on larger architectures, making *MNIST_2x[1024]_ReLU* the most suitable candidate for this analysis. The results, reported in Fig. 7, show that despite the estimation error, PT-LiRPA produces tighter output bounds than α-CROWN, while soundly overapproximating the exact MIP bounds. Similar trends, not reported here for the sake of readability, were consistently observed across all evaluated networks compared with α-CROWN.

We also perform an additional experiment to analyze the impact of the confidence level on the tightness of the bounds. Specifically, we consider the same model of Fig. 7, i.e., *MNIST_2x[1024]_ReLU*, where we have the possibility of employing the exact MIP solver and evaluated a range of increasing confidence levels, namely $1-p \in \{0.8, 0.9, 0.95, 0.99, 0.995, 0.996, 0.997, 0.998, 0.999\}$. For each confidence level, we compute the bounds using α-CROWN, α-CROWN enhanced with our PT-LiRPA, and MIP (which provides the exact bounds). To measure the tightness, we calculate for each of the 10 output nodes the distance between the two bounds computed. For example, if MIP returns output bounds $[-2.2, 3.5]$ and PT-LiRPA returns $[-2.4, 3.7]$, the distance is given by the sum of the absolute differences between the lower and upper bounds, i.e., $|-2.2 + 2.4| + |3.7 - 3.5| = 0.4$. This procedure is repeated for all 10 nodes, collecting, for each confidence level, the mean and standard deviation of these distances.

The results in Fig. 8 clearly show that for moderate confidence levels, i.e., $1 - p \in [0.8, 0.995]$, PT-LiRPA consistently produces sound and tighter bounds compared to α-CROWN. As the confidence level approaches 1 (i.e., as $p \to 0$), the distance between PT-LiRPA and the exact bounds increases, leading to looser bounds than α-CROWN. This trend is perfectly in line with the probabilistic nature of our approach. In fact, considering, for instance, the formula used to compute the probabilistic lower bound, $\hat{l} = Y_1 - \frac{Y_2 - Y_1}{(1-p)^{-a_l} - 1}$ where $a_l \approx \frac{\log(\nu)}{\log\left(\frac{Y_\nu - Y_3}{Y_3 - Y_2}\right)}$

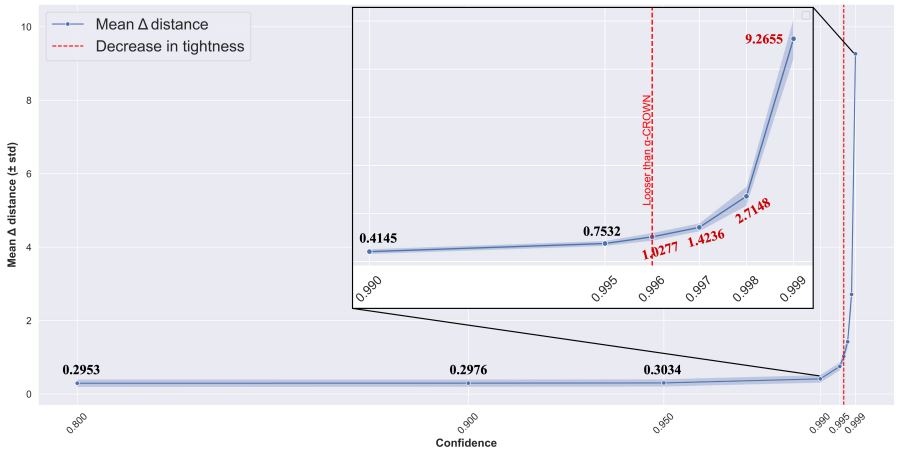

Fig. 8. Mean tightness level of PT-LiRPA for increasing confidence level on *MNIST_2x[1024]_ReLU* model.

($\nu = \lfloor n^{\xi} \rfloor = 350k^{0.85} \simeq 50k$), we observe that as $p \to 0$ (corresponding to requiring very high confidence), the denominator $(1-p)^{-a_l} - 1$ tends to 0 faster than the numerator $(Y_2 - Y_1)$. As a result, the error term $\frac{Y_2 - Y_1}{(1-p)^{-a_l}-1}$ becomes large, and a correspondingly larger margin must be subtracted from the observed minimum. Consequently, the bounds become looser as the confidence level approaches 1. This behavior is mathematically unavoidable since we cannot guarantee arbitrarily high confidence levels while simultaneously maintaining very tight bounds. The plot therefore not only confirms the soundness of our method but also highlights the expected trade-off between confidence and tightness, which is especially relevant for safety-critical applications. For the sake of completeness, we also tested additional models such as *MNIST_2x[1024]_Tanh* and *MNIST_4x[1024]_Sigmoid*. Although MIP was not applicable in these cases, we compared PT-LiRPA only against α-CROWN and observed a similar trend, i.e., a loss of tightness above confidence 0.995, further confirming the generality of our findings.

Answers to Q2. For the second question, we consider the models trained on MNIST and CIFAR datasets, as provided in (H. Zhang, T.-W. Weng, et al. 2018). Hence, we evaluate the performance of our PT-LiRPA-based probabilistic verifier, alongside PROVEN (L. Weng, Chen, et al. 2019), Randomized Smoothing (Cohen et al. 2019), and CROWN (H. Zhang, T.-W. Weng, et al. 2018), by testing for each model 10 random images from the corresponding test datasets. Specifically, we compare the maximum input perturbation ε that can be certified for each method. CROWN (H. Zhang, T.-W. Weng, et al. 2018) serves as the baseline for worst-case robustness certification, as employed in (L. Weng, Chen, et al. 2019). Based on the previous results, we set a sample size of $n = 350k$, $\xi = 0.85$ and $1 - 2mp \geq 0.99$ in our PT-LiRPA-based verifier. This confidence level is consistent with the settings used by L. Weng, Chen, et al. (2019) in their experiments.

Tab.2 reports the results obtained. The *"Worst-case"* column indicates the mean and standard deviation of the maximum ε perturbation tolerated and provably certified using CROWN (H. Zhang, T.-W. Weng, et al. 2018) for that model under consideration in the 10 random images tested, as in (L. Weng, Chen, et al. 2019). Instead, the PROVEN, Rand. Smoothing and PT-LiRPA columns report the certified mean ε and standard deviation we achieve for the same images, using the three probabilistic approaches, sacrificing only a 10^{-2} of confidence. In general, we observe that the PT-LiRPA-based verifier can certify robustness levels up to 3.62 times higher than the worst-case baseline CROWN (H. Zhang, T.-W. Weng, et al. 2018) and up to 3.31 and 2.26 times higher compared to PROVEN (L. Weng, Chen, et al. 2019) and Rand. Smoothing (Cohen et al. 2019), respectively. This demonstrates the significant advantage of our approach over other existing probabilistic methods.

Table 2. Comparison of PT-LiRPA with worst-case bound CROWN (H. Zhang, T.-W. Weng, et al. 2018) and probabilistic approaches PROVEN (L. Weng, Chen, et al. 2019), Randomized Smoothing (Cohen et al. 2019) on different neural networks MNIST and CIFAR models. † results taken from the original paper (L. Weng, Chen, et al. 2019) due to the code's unavailability to reproduce the results.

Certification method Confidence	Worst-case (CROWN) 100%	PROVEN† ≥ 99%	Rand. Smoothing ≥ 99%	CROWN w/ PT-LiRPA ≥ 99%	PT-LiRPA certification bound increase w.r.t. CROWN, PROVEN, Rand. Smoothing
MNIST 2×[1024], ReLU	0.03566±0.011	0.0556	0.0461±0.0106	**0.0558±0.02068**	**1.6X**, **1.004X**, **1.21X**
MNIST 3×[1024], ReLU	0.03112 ± 0.01076	0.03524	0.03452±0.0169	**0.06652 ±0.02501**	**2.14X**, **1.9X**, **1.93X**
MNIST 2×[1024], Tanh	0.01827 ±0.01331	0.02915	0.02301±0.0115	**0.02949 ± 0.02515**	**1.61X**, **1.01X**, **1.28X**
MNIST 3×[1024], Tanh	0.01244 ±0.00468	0.01360	0.01294±0.0073	**0.0257 ± 0.01205**	**2.07X**, **1.89X**, **1.99X**
MNIST 4×[1024], Sigmoid	0.01975 ±0.0111	0.02170	0.02439±0.019	**0.05506±0.04035**	**2.79X**, **2.54X**, **2.26X**
CIFAR 5×[2048], ReLU	0.002412 ± 0.00184	0.00264	0.00778±0.0104	**0.00874 ± 0.00107**	**3.62X**, **3.31X**, **1.12X**
CIFAR 7×[1024], ReLU	0.001984 ± 0.00089	0.00209	**0.006264±0.0061**	0.00471 ± 0.00273	**2.37X**, **2.25X**, 0X

Since the worst-case setting employed in this comparison is one of the first LiRPA-based approaches proposed in the literature, we conduct further analysis on the level of tightness we can achieve with respect to more recent LiRPA approaches such as α-CROWN (Xu, H. Zhang, et al. 2021), β-CROWN (S. Wang, H. Zhang, et al. 2021), and GCP-CROWN (H. Zhang, S. Wang, et al. 2022). For each of these approaches, as well as their probabilistic counterparts based on our PT-LiRPA framework, we compute the mean input perturbation ε that can be certified across 10 random images. As shown in Fig. 9, our probabilistic framework consistently certifies robustness levels at least 1.4 times higher than the worst-case baseline, even when using more recent LiRPA techniques. Notably, the results indicate that as the estimated reachable sets become more precise through over-approximation, the impact of our approach diminishes. Nonetheless, our framework can provide interesting safety information on the model's robustness level even with very tight provable reachable sets.

Finally, to assess the computational overhead introduced by the proposed approach, we perform an ablation study analyzing its impact on the total certification time, considering $n = 350k$ samples and progressively larger network sizes. We highlight that the overall time complexity of the algorithm is polynomial in the network size, specifically $O\left(n \sum_{i=1}^{N} d_i d_{i-1}\right)$, where the term $\sum_{i=1}^{N} d_i d_{i-1}$ represents the complexity for a single sample, where N

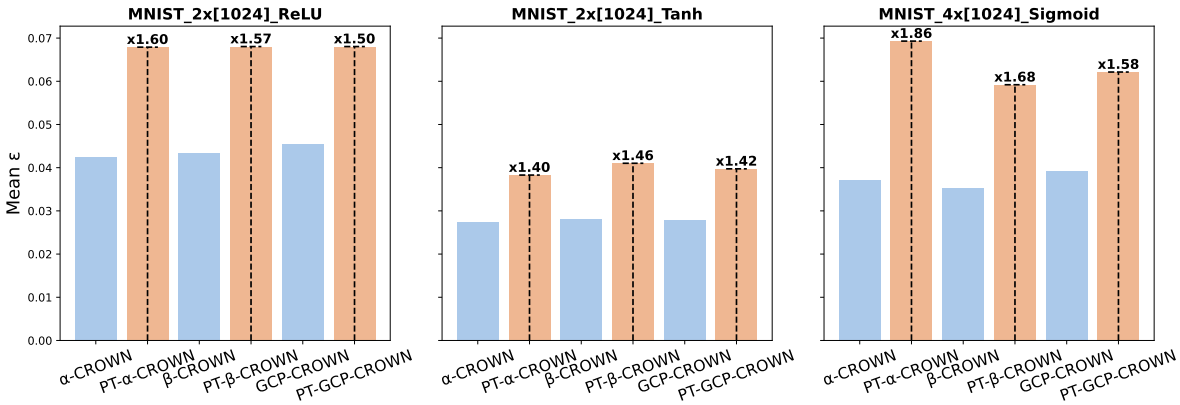

Fig. 9. Comparison PT-LiRPA with worst-case bound CROWN (H. Zhang, T.-W. Weng, et al. 2018), α-CROWN(Xu, H. Zhang, et al. 2021), β-CROWN (S. Wang, H. Zhang, et al. 2021), GCP-CROWN(H. Zhang, S. Wang, et al. 2022) on MNIST_2×[1024]_ReLU, MNIST_2×[1024]_Tanh, and MNIST_4×[1024]_Sigmoid models. On the x-axis, we report the original worst-case method and the corresponding probabilistic version using our PT-LiRPA framework. On the y-axis, we report, for each method, the mean maximum input perturbation ε that can be certified on 10 random images.

is the total number of layers in the network, d_{i-1} is the number of neurons in the preceding layer, and d_i is the number of neurons in the current layer. The product $d_i d_{i-1}$ therefore indicates the number of multiplications required for the processing of layer i. Consequently, the total complexity is proportional to the number of samples multiplied by the cost of a single forward propagation through the entire network. This aspect is well highlighted in the results of Tab. 3, where the time overhead of the sampling-based approach in PT-LiRPA is negligible (i.e., less than one second) in the overall certification time, even when employing a significantly large number of samples, thanks to the GPU acceleration employed in the certification process. Clearly, the total computation time of CROWN enhanced with PT-LiRPA is greater than that of using CROWN alone, as the probabilistic certification of a larger tolerable input perturbation results in a longer verification process.

Table 3. Time comparison between CROWN and CROWN enhanced with PT-LiRPA for the certification of the models in Table 2. The *Total Cert. Time* column reports the overall time required to compute the average ε perturbation that the model can tolerate across 10 random test images. The *# Computations of Interm. Bounds* column indicates how many times the intermediate bound computation procedure is invoked to determine the mean ε, while the *# Samples* column specifies the number of samples used in each instance of intermediate bound computation. Finally, the last two columns present the total overhead and the average time per call of the sampling-based approach used to compute the probabilistically optimal intermediate bounds.

Certification method	**Worst-case (CROWN)**	**CROWN w/ PT-LiRPA**				
	Total Cert. Time	**Total Cert. Time**	**# Computations of Interm. Bounds**	**# Samples**	**Total interm. bounds Comp. time**	**Mean Interm. Bounds Comp. time**
MNIST 2×[1024], ReLU	35.7s	47.4s	253	350k	0.27s	0.001s
MNIST 3×[1024], ReLU	37.73s	54.85s	248	350k	0.34s	0.0014s
MNIST 2×[1024], Tanh	23.02s	36.73s	165	350k	0.22s	0.0013s
MNIST 3×[1024], Tanh	31.47s	59.82s	186	350k	0.26s	0.0014s
MNIST 4×[1024], Sigmoid	40.26s	64.61s	209	350k	0.33s	0.0016s
CIFAR 5×[2048], ReLU	26.3s	82.42s	120	350k	0.48s	0.004s
CIFAR 7×[1024], ReLU	21.02s	73.9s	154	350k	0.37s	0.0024s

Importantly, the time comparison is conducted only against CROWN as the goal of this experiment is to show that the additional cost introduced by PT-LiRPA is negligible in the overall verification time, while still producing tighter output bounds. Importantly, once the LiRPA baseline is fixed (e.g., α-CROWN, β-CROWN, GCP-CROWN), the subsequent linearization procedure is identical whether using the original method or our PT-LiRPA variant. The only difference lies in the way intermediate bounds are computed, which are then used to construct the diagonal matrices and bias terms. Consequently, we argue that measuring the overhead against CROWN is representative, since the additional cost introduced by PT-LiRPA does not depend on which LiRPA baseline is employed.

Answers to Q3. To answer the last question, we integrate our PT-LiRPA in the α, β-CROWN toolbox (https://github.com/Verified-Intelligence/alpha-beta-CROWN) and perform a final experiment on different benchmarks of the VNN-COMP 2022 and 2023 (Brix et al. 2023; Müller et al. 2022). We point out that the comparison between probabilistic and provable verifiers is performed to have a ground truth (when possible), and to highlight the valuable help that probabilistic approaches can have in solving challenging instances to be verified.
To keep the paper self-contained, we report below a brief overview of the selected benchmarks.

- *ACAS xu* (K. D. Julian et al. 2016; Katz et al. 2017) benchmark 2023: includes ten properties evaluated across 45 neural networks designed to provide turn advisories for aircraft to prevent collisions. Each neural network consists of 300 neurons distributed over six layers, using ReLU activation functions. The networks take five inputs representing the aircraft's state and produce five outputs, with the advisory determined by the minimum output value. Here, we verified only property 3, which returns unsafe if the clear of conflict (COC) output is minimal, with a max computation time of 116s.

- *TllVerifyBench* benchmark 2023: this benchmark features Two-Level Lattice (TLL) neural networks with two inputs and one single output. These models are then transformed into MLP ReLU networks where the output properties consist of a randomly generated real number and a randomly generated inequality direction to be verified. Here we verify all 32 instances of the VNN-COMP 2023 with a timeout of 600s for each property.
- *CIFAR_biasfield* benchmark 2022: this benchmark focuses on verifying a Cifar-10 network under bias field perturbations. These perturbations are modeled by creating augmented networks that reduce the input space to just 16 parameters. For each image to be verified, a distinct bias field transform network is generated, consisting of a fully connected transform layer followed by the Cifar CNN with 8 convolutional layers with ReLU activations. Each bias field transform network has 363k parameters and 45k nodes. Here, we test all 72 properties with a timeout set to 300s for each one.
- *TinyImageNet* benchmark 2022: consists of CIFAR100 image classification ($56 \times 56 \times 3$) with Residual Neural Networks (ResNet). Here, we consider the medium network size composed of 8 residual blocks, 17 convolutional layers, and 2 linear layers. For *TinyImageNet-ResNet-medium*, we verify all 24 properties with a timeout of 200 seconds for each property.

In general, we selected benchmarks where the state-of-the-art α, β-CROWN method is unable to solve some of the instances within the time constraints. This set of experiments aims to confirm our hypothesis regarding the effectiveness of having tighter estimated reachable sets for verification purposes. In detail, our intuition is that even though our procedure requires a little initial overhead, with tighter estimated reachable sets, we can achieve more precise final output bounds, potentially reducing the cases where the verification approach can not make a decision and must resort to a split in the BaB process, thus resulting in more efficient overall verification time. Before starting to verify these instances, we explore the effect of varying incremental sample sizes for a fixed $\xi = 0.85$ on the computation of estimated reachable sets with neural networks of significant size.

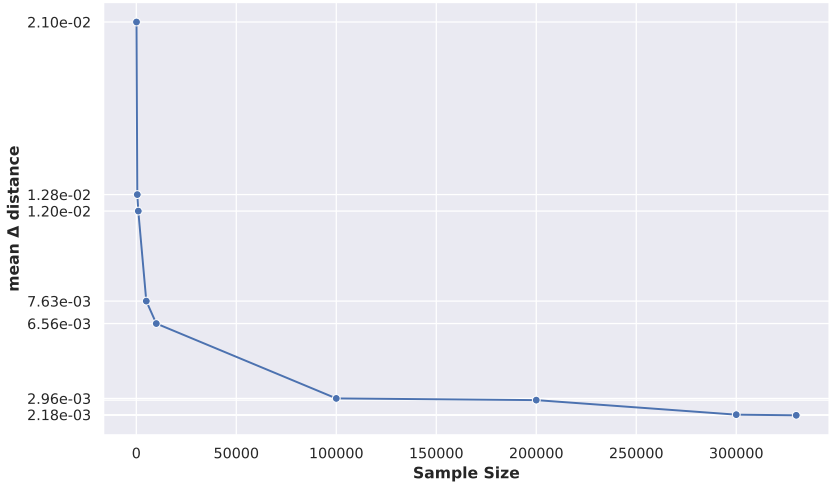

Fig. 10. Estimated reachable sets at convergence for the increasing sample size in *CIFAR_biasfield* benchmark. y-axis reports the mean distance between estimated reachable sets using $350k$ samples (as reference) and the one using $[100, 500, 1k, 5k, 10k, 100k, 200k, 300k, 330k]$, respectively.

In detail, we focus on the *CIFAR_biasfield* benchmark and set a confidence level of $1 - 2mp \geq 99\%$. In detail, we compute the mean distance between estimated reachable sets using $350k$ samples as reference and the one

Table 4. Results on VNN-COMP 2022-2023 benchmarks. Results marked in **bold** report improved performance in terms of verified accuracy (% sat instances/all instances) and total verification time for the specific benchmark tested, w.r.t. a worst-case verification approach.

Benchmark	Method	Confidence	Verified accuracy	#safe (unsat)	#unsafe (sat)	#unkwown	Tot verification time
ACASxu	α,β-CROWN	100%	93.33%	42	3	0	26s
	α,β-CROWN w/PT-LiRPA	$\geq$ 99%	93.33%	42	3	0	**16.37s**
tllVerifyBench	α,β-CROWN	100%	46.875%	15	17	0	90.2s
	α,β-CROWN w/PT-LiRPA	$\geq$ 99%	46.875%	15	17	0	92s
CIFAR_biasfield	α,β-CROWN	100%	95.83%	69	1	2	1553.5s
	α,β-CROWN w/PT-LiRPA	$\geq$ 99%	**98.61%**	**71**	1	**0**	**408.7s**
CIFAR_tinyimagenet	α,β-CROWN	100%	62.5%	15	3	6	1429.6s
	α,β-CROWN w/PT-LiRPA	$\geq$ 99%	**87.5%**	**21**	3	**0**	**425.6s**

using progressively increasing the sample size until the difference between successive estimated reachable sets exceeds the threshold of $\Delta = 0.001$. Specifically, we begin with 100 samples and progressively increase the sample size until the difference between successive estimated reachable sets exceeds the threshold. Our results, detailed in Fig. 10, indicate that stable estimated reachable sets, in this scenario, can be obtained with sample sizes ranging from $300k$ to $350k$ as the mean distance between estimated bounds is strictly less than $\Delta = 0.001$. Hence, employing a sample size of $350k$ samples results in a valid choice even for considerably large networks. We recall once again that propagating a large number of samples, such as $350k$, requires a negligible computational effort and time (as shown in the results of Tab. 3) due to batch processing and GPU acceleration. The principal limitation arises from GPU memory capacity, since larger sample sizes may increase the risk of memory errors relative to CPU-based propagation.

In Tab. 4 we report our results on the VNN-COMP, where we consider an increased difficulty for the verification process. We start with the simpler benchmark *ACASxu* (K. D. Julian et al. 2016; Katz et al. 2017), and we test property 3. This property is particularly interesting as it holds for 42 of the 45 models tested, thus allowing us to verify the improvement in terms of time and verification accuracy. In the first row of Tab. 4, we can notice that by sacrificing only a 0.01% of confidence, α, β-CROWN enhanced with PT-LiRPA achieves the same verified accuracy in less verification time, thus confirming our intuition. Interestingly, we observe that tighter bounds are not always beneficial in general. Specifically, in cases where a PGD attack succeeds despite loose bounds, using tighter bounds does not lead to further improvements. Additionally, in some scenarios, less accurate bounds from vanilla LiRPA methods could be quickly refined by BaB, still resulting in efficient verification time. This is exemplified by the *tllVerifyBench* experiments, where even sacrificing a 0.01% of confidence, PT-LiRPA produced tight estimated reachable sets but achieved the same verified accuracy with a minor overhead in bounds computation.
Crucially, the real benefit of our PT-LiRPA arises on more challenging verification benchmarks such as *CIFAR_biasfield* and *CIFAR_tinyimagenet*. Both these benchmarks are image-based verification tasks and thus allow us to show the scalability and the impact of tightened estimated reachable sets on large networks using the proposed approach. Crucially, in these two last benchmarks, sacrificing a 0.01% of confidence, we obtain significant improvements in verification results with respect to worst-case α, β-CROWN. In detail, in both *CIFAR_biasfield* and *CIFAR_tinyimagenet*, we achieved higher verified accuracy without incurring any *unknown* answer and with significantly less verification time. These final results demonstrate the effectiveness and the potential impact of using PT-LiRPA for verification purposes, showing the advantage of incorporating estimated reachable sets in handling challenging instances that are difficult to solve with provable solvers.

6 Assumptions and Limitations

The proposed PT-LiRPA framework builds on several assumptions that define its applicability and theoretical guarantees. Our probabilistic framework relies on uniform random sampling within the perturbation region $C_{\mathbf{x}_0,\epsilon}$ to estimate probabilistically tight reachable sets. Consequently, our theoretical probabilistic guarantees, derived from Wilks (1942)'s tolerance limit theorem and extended using extreme value theory (Haan and Ferreira 2006), hold under the assumption that the samples are independent and representative of the true input distribution within C.

Importantly, by accepting robustness certificates that hold for a fraction R of the perturbation region, the theoretical and practical tool derived from Wilks (1942)'s results, i.e., Theorem 3.4, remains a valuable solution, as it provides a closed-form expression to compute the number of samples required for any desired confidence level ψ and coverage ratio R. To address this limitation and extend the analysis to probabilistic certificates valid over the entire perturbation region, thus aligning with the probabilistic verification literature, we further based our theoretical derivation on extreme value theory and the results of De Haan (1981). In this case, our derivation in Theorem 3.7 provides, for any choice of sample size n and confidence error p, a guarantee that holds for all input values $\boldsymbol{x} \in C$. However, unlike Wilks' approach, EVT does not yield a closed-form expression to determine the minimum number of samples required to achieve a desired precision and confidence level. Consequently, the balance between probabilistic soundness and the tightness of the resulting output bounds becomes more empirical, as confirmed in our experiments. In fact, the tightness of the computed bounds and the computational efficiency of the method depend on the chosen sample size n, and the estimate of the tail distribution $\nu = \lfloor n^{\xi} \rfloor$, which must balance verification accuracy and computational cost.

Despite these limitations, we argue that PT-LiRPA complements deterministic verification methods by offering practical, quantifiable robustness guarantees for challenging instances where worst-case formal verification is either overly conservative or computationally infeasible.

7 Conclusion

We introduced PT-LiRPA, a novel probabilistic framework that combines LiRPA-based formal verification of deep neural networks approaches with a sampling-based technique. We provide a rigorous theoretical derivation of the correctness of our approach, complementing, for the first time, statistical results on the tolerance limit, with qualitative bounds on the error magnitude of a sampling-based approach employed to estimate reachable set domains. Our approach provides tighter linear bounds, significantly improving both the accuracy and verification efficiency while maintaining provable probabilistic guarantees on the soundness of the verification result. Empirical results demonstrate that PT-LiRPA outperforms related probabilistic methods, particularly in robustness certification, decreasing the confidence in the result by infinitesimal amounts. Additionally, we show the potential of our probabilistic approach for verifying challenging instances where the formal approaches fail.

Acknowledgments

This work has been supported by PNRR MUR project PE0000013-FAIR.

References

G. Amir, D. Corsi, R. Yerushalmi, L. Marzari, D. Harel, A. Farinelli, and G. Katz. 2023. "Verifying learning-based robotic navigation systems." In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 607–627.

F. Archetti and F. Schoen. 1984. "A survey on the global optimization problem: general theory and computational approaches." *Annals of Operations Research*, 1, 87–110.

M. Balunovic, M. Baader, G. Singh, T. Gehr, and M. Vechev. 2019. "Certifying geometric robustness of neural networks." *Advances in Neural Information Processing Systems*, 32.

T. Belkhouja and J. R. Doppa. 2022. "Adversarial framework with certified robustness for time-series domain via statistical features." *Journal of Artificial Intelligence Research*, 73, 1435–1471.

L. Berrada, S. Dathathri, K. Dvijotham, R. Stanforth, R. R. Bunel, J. Uesato, S. Gowal, and M. P. Kumar. 2021. "Make sure you're unsure: A framework for verifying probabilistic specifications." *Advances in Neural Information Processing Systems*, 34, 11136–11147.

Y. Biktairov and J. Deshmukh. 2023. "SOL: Sampling-based Optimal Linear bounding of arbitrary scalar functions." *Advances in Neural Information Processing Systems*, 36, 33161–33173.

D. Boetius, S. Leue, and T. Sutter. 2024. *Probabilistic Verification of Neural Networks using Branch and Bound*. (2024).

C. Brix, S. Bak, C. Liu, and T. T. Johnson. 2023. *The fourth international verification of neural networks competition (VNN-COMP 2023): Summary and results*. (2023).

R. Bunel, J. Lu, I. Turkaslan, P. H. Torr, P. Kohli, and M. P. Kumar. 2020. "Branch and bound for piecewise linear neural network verification." *Journal of Machine Learning Research*, 21, 42, 1–39.

S. Carr, N. Jansen, and U. Topcu. 2021. "Task-aware verifiable RNN-based policies for partially observable Markov decision processes." *Journal of Artificial Intelligence Research*, 72, 819–847.

J. Cohen, E. Rosenfeld, and Z. Kolter. 2019. "Certified adversarial robustness via randomized smoothing." In: *international conference on machine learning*. PMLR, 1310–1320.

N. Couellan. 2021. "Probabilistic robustness estimates for feed-forward neural networks." *Neural networks*, 142, 138–147.

L. De Haan. 1981. "Estimation of the minimum of a function using order statistics." *Journal of the American Statistical Association*, 76, 374, 467–469.

K. Dvijotham, M. Garnelo, A. Fawzi, and P. Kohli. 2018. *Verification of deep probabilistic models*. (2018).

R. Ehlers. 2017. "Formal verification of piece-wise linear feed-forward neural networks." In: *Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15*. Springer, 269–286.

S. Gao, S. Kong, and E. M. Clarke. 2013. "dReal: An SMT solver for nonlinear theories over the reals." In: *International conference on automated deduction*. Springer, 208–214.

L. Haan and A. Ferreira. 2006. *Extreme value theory: an introduction*. Vol. 3. Springer.

D. Hendrycks and K. Gimpel. 2016. *Gaussian error linear units (gelus)*. (2016).

K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer. 2016. "Policy compression for aircraft collision avoidance systems." In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 1–10.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. 2017. "Reluplex: An efficient SMT solver for verifying deep neural networks." In: *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 97–117.

S. Kotha, C. Brix, J. Z. Kolter, K. Dvijotham, and H. Zhang. 2023. "Provably bounding neural network preimages." *Advances in Neural Information Processing Systems*, 36, 80270–80290.

C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer, et al.. 2021. "Algorithms for verifying deep neural networks." *Foundations and Trends® in Optimization*, 4, 3-4, 244–404.

A. Lomuscio and L. Maganti. 2017. *An approach to reachability analysis for feed-forward relu neural networks*. (2017).

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. 2018. "Towards Deep Learning Models Resistant to Adversarial Attacks." In: *International Conference on Learning Representations (ICLR)*.

R. Mangal, A. V. Nori, and A. Orso. 2019. "Robustness of neural networks: A probabilistic and practical approach." In: *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 93–96.

L. Marzari, F. Cicalese, A. Farinelli, C. Amato, and E. Marchesini. Nov. 2025. "Verifying Online Safety Properties for Safe Deep Reinforcement Learning." *ACM Trans. Intell. Syst. Technol.*, 17, 1, Article 3, (Nov. 2025), 27 pages. doi:10.1145/3770068.

L. Marzari, D. Corsi, F. Cicalese, and A. Farinelli. 2023. "The #DNN-verification problem: counting unsafe inputs for deep neural networks." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 217–224.

L. Marzari, D. Corsi, E. Marchesini, and A. Farinelli. 2022. "Curriculum learning for safe mapless navigation." In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 766–769.

L. Marzari, D. Corsi, E. Marchesini, A. Farinelli, and F. Cicalese. 2024. "Enumerating safe regions in deep neural networks with provable probabilistic guarantees." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 19. Vol. 38, 21387–21394.

L. Marzari, P. L. Donti, C. Liu, and E. Marchesini. 2025. "Improving Policy Optimization via ϵ-Retrain." In: *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 1464–1472. doi:10.5555/3709347.3743780.

L. Marzari, A. Pore, D. Dall'Alba, G. Aragon-Camarasa, A. Farinelli, and P. Fiorini. 2021. "Towards Hierarchical Task Decomposition using Deep Reinforcement Learning for Pick and Place Subtasks." In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 640–645.

P. Morettin, A. Passerini, and R. Sebastiani. 2024. *A Unified Framework for Probabilistic Verification of AI Systems via Weighted Model Integration*. (2024).

M. N. Müller, C. Brix, S. Bak, C. Liu, and T. T. Johnson. 2022. *The third international verification of neural networks competition (VNN-COMP 2022): Summary and results*. (2022).

K. O'Shea and R. Nash. 2015. *An introduction to convolutional neural networks*. (2015).

B. Paulsen and C. Wang. 2022. "LinSyn: Synthesizing tight linear bounds for arbitrary neural network activation functions." In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 357–376.

M. Pautov, N. Tursynbek, M. Munkhoeva, N. Muravev, A. Petiushko, and I. Oseledets. 2022. "CC-Cert: A probabilistic approach to certify general robustness of neural networks." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 7. Vol. 36, 7975–7983.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. 2018. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations." In: *Proceedings of Robotics: Science and Systems (RSS)*.

P. Ramachandran, B. Zoph, and Q. V. Le. 2017. *Searching for activation functions*. (2017).

G. Singh, T. Gehr, M. Püschel, and M. Vechev. 2019. "An abstract domain for certifying neural networks." *Proceedings of the ACM on Programming Languages*, 3, POPL, 1–30.

V. Sivaramakrishnan, K. C. Kalagarla, R. Devonport, J. Pilipovsky, P. Tsiotras, and M. Oishi. 2024. *SAVER: A Toolbox for Sampling-Based, Probabilistic Verification of Neural Networks*. (2024).

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2013. *Intriguing properties of neural networks*. (2013).

L. Tai, G. Paolo, and M. Liu. 2017. "Virtual-to-real DRL: Continuous control of mobile robots for mapless navigation." In: *IROS*.

V. Tjeng, K. Xiao, and R. Tedrake. 2017. "Evaluating robustness of neural networks with mixed integer programming." In: *International Conference on Learning Representations*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. "Attention is all you need." *Advances in neural information processing systems*, 30.

S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. 2018. "Formal security analysis of neural networks using symbolic intervals." In: *27th USENIX Security Symposium (USENIX Security 18)*, 1599–1614.

S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter. 2021. "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification." *Advances in Neural Information Processing Systems*, 34, 29909–29921.

S. Webb, T. Rainforth, Y. W. Teh, and M. P. Kumar. 2018. *A statistical approach to assessing neural network robustness*. (2018).

T. Wei, H. Hu, L. Marzari, K. S. Yun, P. Niu, X. Luo, and C. Liu. 2025. "Modelverification. jl: a comprehensive toolbox for formally verifying deep neural networks." In: *Proceedings of the 37th International Conference on Computer Aided Verification*.

L. Weng, P.-Y. Chen, L. Nguyen, M. Squillante, A. Boopathy, I. Oseledets, and L. Daniel. 2019. "PROVEN: Verifying robustness of neural networks with a probabilistic approach." In: *International Conference on Machine Learning*. PMLR, 6727–6736.

L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. 2018. "Towards fast computation of certified robustness for relu networks." In: *International Conference on Machine Learning*. PMLR, 5276–5285.

T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. 2018. "Evaluating the robustness of neural networks: An extreme value theory approach." In.

S. S. Wilks. 1942. "Statistical prediction with special reference to the problem of tolerance limits." *The annals of mathematical statistics*, 13, 4, 400–409.

H. Wu et al.. 2024. "Marabou 2.0: a versatile formal analyzer of neural networks." In: *International Conference on Computer Aided Verification*. Springer, 249–264.

K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh. 2020. "Automatic perturbation analysis for scalable certified robustness and beyond." *Advances in Neural Information Processing Systems*, 33, 1129–1141.

K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C.-J. Hsieh. 2021. "Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers." In: *International Conference on Learning Representations*.

Z. Xue, S. Liu, Z. Zhang, Y. Wu, and M. Zhang. 2023. "A Tale of Two Approximations: Tightening Over-Approximation for DNN Robustness Verification via Under-Approximation." In: *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (ISSTA 2023), 1182–1194.

H. Zhang, S. Wang, K. Xu, L. Li, B. Li, S. Jana, C.-J. Hsieh, and J. Z. Kolter. 2022. "General cutting planes for bound-propagation-based neural network verification." *Advances in neural information processing systems*, 35, 1656–1670.

H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. 2018. "Efficient neural network robustness certification with general activation functions." *Advances in neural information processing systems*, 31.

X. Zhang, B. Wang, and M. Kwiatkowska. 2024. "Provable preimage under-approximation for neural networks." In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 3–23.

Y. Zhang, Y. Tang, W. Ruan, X. Huang, S. Khastgir, P. Jennings, and X. Zhao. 2025. "ProTIP: Probabilistic robustness verification on text-to-image diffusion models against stochastic perturbation." In: *European Conference on Computer Vision*. Springer, 455–472.

Received 21 October 2025; accepted 24 November 2025