Enumerating Safe Regions in Deep Neural Networks with Provable Probabilistic Guarantees

Luca Marzari[1], **Davide Corsi**[1], **Enrico Marchesini**[2], **Alessandro Farinelli**[1], **Ferdinando Cicalese**[1]
[1]Department of Computer Science, University of Verona, Italy
[2]Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, USA
{luca.marzari, davide.corsi, alessandro.farinelli, ferdinando.cicalese}@univr.it
emarche@mit.edu

Abstract

Identifying safe areas is a key point to guarantee trust for systems that are based on Deep Neural Networks (DNNs). To this end, we introduce the *AllDNN-Verification* problem: given a safety property and a DNN, enumerate the set of all the regions of the property input domain which are safe, i.e., where the property does hold. Due to the #P-hardness of the problem, we propose an efficient approximation method called ϵ-`ProVe`. Our approach exploits a controllable underestimation of the output reachable sets obtained via statistical prediction of tolerance limits, and can provide a tight —with provable probabilistic guarantees— lower estimate of the safe areas. Our empirical evaluation on different standard benchmarks shows the scalability and effectiveness of our method, offering valuable insights for this new type of verification of DNNs.

Introduction

Deep Neural Networks (DNNs) have emerged as a groundbreaking technology revolutionizing various fields ranging from autonomous navigation (Tai, Paolo, and Liu 2017; Marzari et al. 2022) to image classification (O'Shea and Nash 2015) and robotics for medical applications (Corsi et al. 2023). However, while DNNs can perform remarkably well in different scenarios, their reliance on massive data for training can lead to unexpected behaviors and vulnerabilities in real-world applications. In particular, DNNs are often considered ”black-box” systems, meaning their internal representation is not fully transparent. A crucial DNNs weakness is the vulnerability to adversarial attacks (Szegedy et al. 2013; Amir et al. 2023), wherein small, imperceptible modifications to input data can lead to wrong and potentially catastrophic decisions when deployed.

To this end, Formal Verification (FV) of DNNs (Katz et al. 2017; Liu et al. 2021) holds great promise to provide assurances on the safety aspect of these functions before the actual deployment in real scenarios. In detail, the decision version of the *DNN-Verification* problem takes as input a trained DNN $\mathcal{N}$ and a safety property, typically expressed as an input-output relationship for $\mathcal{N}$, and aims at determining whether there exists at least an input configuration which results in a violation of the safety property. It is crucial to point out that due to the fact that the DNN's input is typically defined in a continuous domain any empirical evaluation of a safety property cannot rely on testing all the (infinitely many) possible input configurations. In contrast, FV can provide provable assurances on the DNNs' safety aspect. However, despite the considerable advancements made by DNN-verifiers over the years (Katz et al. 2019; Wang et al. 2021; Liu et al. 2021), the binary result (*safe* or *unsafe*) provided by these tools is generally not sufficient to gain a comprehensive understanding of these functions. For instance, when comparing two neural networks and employing an FV tool that yields an *unsafe* answer for both (i.e., indicating the existence of at least one violation point), we cannot distinguish whether one model exhibits only a small area of violation around the identified counterexample, while the other may have multiple and widespread violation areas.

To overcome this limitation, a quantitative variant of FV, asking for the number of violation points, has been proposed and analyzed, first in (Baluta et al. 2019) for the restricted class of Binarized Neural Networks (BNNs) and more recently in (Marzari et al. 2023) for general DNNs. Following (Marzari et al. 2023) we will henceforth refer to such counting problem as *#DNN-Verification*. Due to the #P-hardness of the *#DNN-Verification*, both studies in (Baluta et al. 2019; Marzari et al. 2023), focus on efficient approximate solutions, which allow the resolution of large-scale real-world problems while providing provable (probabilistic) guarantees regarding the computed count.

Solutions to the *#DNN-Verification* problem allow to estimate the probability that a DNN violates a given property but they do not provide information on the actual input configurations that are safe or violations for the property of interest.

On the other hand, knowledge of the distribution of safe and unsafe areas in the input space is a key element to devise approaches that can enhance the safety of DNNs, e.g., by patching unsafe areas through re-training.

To this aim, we introduce the *AllDNN-Verification* problem , which corresponds to computing the set of all the areas that do not result in a violation for a given DNN and a safety property (i.e., enumerating all the safe areas of a property's input domain). The *AllDNN-Verification* is at least as hard as *#DNN-Verification*, i.e., it is easily shown to be #P-Hard.

Hence, we propose ϵ−ProVe, an approximation approach that provides provable (probabilistic) guarantees on the returned areas. ϵ−ProVe is built upon interval analysis of the DNN output reachable set and the iterative refinement approach (Wang et al. 2018b), enabling efficient and reliable enumeration of safe areas.[1]

Notice that state-of-the-art FV methods typically propose an over-approximated output reachable set, thereby ensuring the soundness of the result. Nonetheless, the relaxation of the nonlinear activation functions employed to compute the over-approximate reachable set has non-negligible computational demands. In contrast, ϵ−ProVe provides a scalable solution based on an underestimation of the output reachable set that exploits the *Statistical Prediction of Tolerance Limits* (Wilks 1942; Porter 2019). In particular, we demonstrate how, with a confidence α, our underestimation of the reachable set computed with n random input configurations sampled from the initial property's domain A is a correct output reachable set for at least a fraction R of an indefinitely large further sample of points. Broadly speaking, this result tells us that if all the input configurations obtained in a random sample produce an output reachable set that does not violate the safety property (i.e., a *safe* output reachable set) then, with probability α, at least a subset of A of size $R \cdot |A|$ is safe (i.e., R is a lower bound on the safe rate in A with confidence α). In summary, the main contributions of this paper are the following:

- We initiate the study of the *AllDNN-verification* problem, the enumeration version of the DNN-Verification
- Due to the #P-hardness of the problem, we propose ϵ−ProVe a novel approximation method to obtain a provable (probabilistic) lower bound of the safe zones within a given property's domain.
- We evaluate our approach on FV standard benchmarks, showing that ϵ−ProVe is scalable and effective for real-world scenarios.

Preliminaries

In this section, we discuss existing FV approaches and related key concepts on which our approach is based. In contrast to the standard robustness and adversarial attack literature (Carlini and Wagner 2017; Madry et al. 2017; Zhang et al. 2022b), FV of DNNs seeks formal guarantees on the safety aspect of the neural network given a specific input domain of interest. Broadly speaking, if a DNN-Verification tool states that the region is provably verified, this implies that there are no adversarial examples – violation points – in that region. We recall in the next section the formal definition of the satisfiability problem for *DNN-Verification* (Katz et al. 2017).

### DNN-Verification

In the *DNN-Verification*, we have as input a tuple $\mathcal{T} = \langle \mathcal{N}, \mathcal{P}, \mathcal{Q} \rangle$, where $\mathcal{N}$ is a DNN, $\mathcal{P}$ is precondition on the input, and $\mathcal{Q}$ a postcondition on the output. In particular, $\mathcal{P}$ denotes a particular input domain or region for which we require a particular postcondition $\mathcal{Q}$ to hold on the output of $\mathcal{N}$. Since we are interested in discovering a possible counterexample, $\mathcal{Q}$ typically encodes the negation of the desired behavior for $\mathcal{N}$. Hence, the possible outcomes are SAT if there exists an input configuration that lies in the input domain of interest, satisfying the predicate $\mathcal{P}$, and for which the DNN satisfies the postcondition $\mathcal{Q}$, i.e., at least one violation exists in the considered area, UNSAT otherwise.

[1]We point out that the *AllDNN-Verification* can also be defined to compute the set of unsafe regions. For better readability, we will only focus on safe regions. The definition and the solution proposed are directly derivable also when applied to unsafe areas.

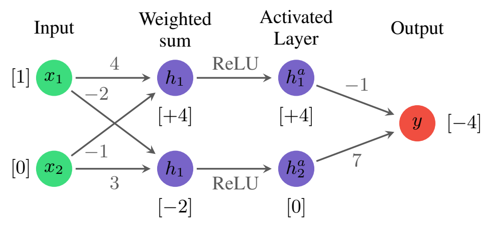

Figure 1: A counterexample for a toy *DNN-Verification* problem.

To provide the reader with a better intuition on the *DNN-Verification* problem, we discuss a toy example.

Example 1. *(DNN-Verification) Suppose we want to verify that for the toy DNN $\mathcal{N}$ depicted in Fig. 1 given an input vector $x = (x_1, x_2) \in [0,1] \times [0,1]$, the resulting output should always be $y \geq 0$. We define $\mathcal{P}$ as the predicate on the input vector $x = (x_1, x_2)$ which is true iff $x \in [0,1] \times [0,1]$, and $\mathcal{Q}$ as the predicate on the output y which is true iff $y = \mathcal{N}(x) < 0$, that is, we set $\mathcal{Q}$ to be the negation of our desired property. As reported in Fig. 1, given the vector $x = (1,0)$ we obtain $y < 0$, hence the verification tool returns a* SAT *answer, meaning that a specific counterexample exists and thus the original safety property does not hold.*

#DNN-Verification

Despite the provable guarantees and the advancement that formal verification tools have shown in recent years (Liu et al. 2021; Katz et al. 2019; Wang et al. 2021; Zhang et al. 2022a), the binary nature of the result of the *DNN-Verification* problem may hide additional information about the safety aspect of the DNNs. To address this limitation in (Marzari et al. 2023) the authors introduce the *#DNN-Verification*, i.e., the extension of the decision problem to its counting version. In this problem, the input is the same as the decision version, but we denote as $\Gamma(\mathcal{T})$ the set of *all* the input configurations for $\mathcal{N}$ satisfying the property defined by $\mathcal{P}$ and $\mathcal{Q}$, i.e.

$$\Gamma(\mathcal{T}) = \left\{ x \;\middle|\; \mathcal{P}(x) \wedge \mathcal{Q}(\mathcal{N}(x)) \right\} \tag{1}$$

Then, the *#DNN-Verification* consists of computing $|\Gamma(\mathcal{T})|$.

The approach (reported in Fig.2) solves the problem in a sound and complete fashion where any state-of-the-art FV tool for the decision problem can be employed to check each

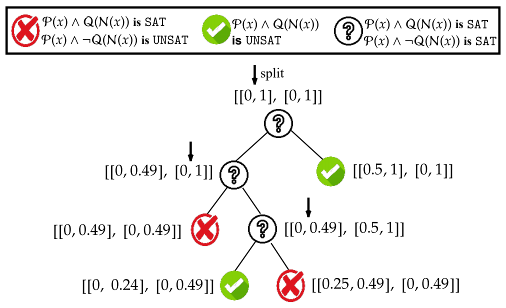
$\mathcal{P}(x) \wedge \mathcal{Q}(\mathcal{N}(x))$ is SAT
$\mathcal{P}(x) \wedge \neg\mathcal{Q}(\mathcal{N}(x))$ is UNSAT
$\mathcal{P}(x) \wedge \mathcal{Q}(\mathcal{N}(x))$ is UNSAT
$\mathcal{P}(x) \wedge \mathcal{Q}(\mathcal{N}(x))$ is SAT
$\mathcal{P}(x) \wedge \neg\mathcal{Q}(\mathcal{N}(x))$ is SAT
split
[[0, 1], [0, 1]]
[[0, 0.49], [0, 1]]
[[0.5, 1], [0, 1]]
[[0, 0.49], [0, 0.49]]
[[0, 0.49], [0.5, 1]]
[[0, 0.24], [0, 0.49]]
[[0.25, 0.49], [0, 0.49]]

Figure 2: Explanatory image execution of exact count for a particular $\mathcal{N}$ and safety property.

node of the Branch-and-Bound (Bunel et al. 2018) tree recursively. In detail, each node produces a partition of the input space into two equal parts as long as it contains both a point that violates the property and a point that satisfies it. The leaves of this recursion tree procedure correspond to partitioning the input space into parts where we have either complete violations or safety. Hence, the provable count of the safe areas is easily computable by summing up the cardinality of the subinput spaces in the leaves that present complete safety. Since our setting is in the continuum, the number of points in any non-empty set is infinite. Hence, we consider the cardinality as a proxy for the volume of the corresponding set. Nonetheless, if we assume some discretization of the space (to the maximum resolution allowed by the machine precision), $\Gamma(\mathcal{T})$ becomes a finite countable set.

Clearly, by using this method, it is possible to exactly count (and even to enumerate) the safe points. However, due to the necessity of solving a *DNN-Verification* instance at each node (an intractable problem that might require exponential time), this approach becomes soon unfeasible and struggles to scale on real-world scenarios. In fact, it turns out that under standard complexity assumption, no efficient and scalable approach can return the exact set of areas in which a DNN is provably safe (as detailed in the next section).

To address this concern, after formally defining the *AllDNN-Verification* problem and its complexity, we propose first of all a relaxation of the problem, and subsequently, an approximate method that exploits the analysis of underestimated output reachable sets obtained using statistical prediction of tolerance limits (Wilks 1942; Porter 2019) and provides a tight underapproximation of the safe areas with strong probabilistic guarantees.

The *AllDNN-Verification* Problem

The *AllDNN-Verification* problem asks for the set of all the safe points for a particular tuple $\langle \mathcal{N}, \mathcal{P}, \mathcal{Q} \rangle$. Formally:

Definition 1 (*AllDNN-Verification Problem*).
Input*: A tuple $\mathcal{T} = \langle \mathcal{N}, \mathcal{P}, \mathcal{Q} \rangle$.*
Output*: the set of safe points $\Gamma(\mathcal{T})$, as given in (1).*

Considering the example of Fig. 2 solving the *AllDNN-Verification* problem for the safe areas consists in returning the set:
$$\Gamma = \Big\{ [[0.5, 1] \times [0, 1]] \cup [[0, 0.24] \times [0, 0.49]] \Big\}.$$

Hardness of *AllDNN-Verification*

From the #P-hardness of the *#DNN-Verification* problem proved in (Marzari et al. 2023) and the fact that exact enumeration also provides exact counting it immediately follows that the *AllDNN-Verification* is *#P-hard*, which essentially states that no polynomial algorithm is expected to exists for the *AllDNN-Verification* problem.

ϵ-ProVe: a Provable (Probabilistic) Approach

In view of the structural scalability issue of any solution to the *AllDNN-Verification* problem, due to its #P-hardness, we propose to resort to an approximate solution. More precisely, we define the following approximate version of the *AllDNN-Verification* problem:

Definition 2 (*ϵ-Rectilinear Under-Approximation of safe areas for DNN (ϵ-RUA-DNN)*).
Input*: A tuple $\mathcal{T} = \langle \mathcal{N}, \mathcal{P}, \mathcal{Q} \rangle$.*
Output*: a family $\mathcal{R} = \{r_1, \ldots, r_m\}$ of disjoint rectilinear ϵ-bounded hyperrectangles such that $\bigcup_i r_i \subseteq \Gamma(\mathcal{T})$ and $|\Gamma(\mathcal{T}) \setminus \bigcup_i r_i|$ is minimum.*

A *rectilinear ϵ-bounded hyperrectangle* is defined as the cartesian product of intervals of size at least ϵ. Moreover, for $\epsilon > 0$, we say that a rectilinear hyperrectangle $r = \times_i [\ell_i, u_i]$ is *ϵ-aligned* if for each i, both extremes ℓ_i and u_i are a multiple of ϵ.

The rationale behind this new formulation of the problem is twofold: on the one hand, we are relaxing the request for the exact enumeration of safe points—in fact, as argued in (Karp and Luby 1985) due to the #P-hardness proof (from #3-SAT), even guaranteeing a constant approximation to $|\Gamma(\mathcal{T})|$ by a deterministic polynomial time algorithm is not possible unless $P = NP$; on the other hand, we are requiring that the output is more concisely representable by means of hyperrectangles of *some significant* size.

Note that for $\epsilon \rightarrow 0$, *ϵ-RUA-DNN* and *AllDNN-Verification* become the same problem. More generally, whenever the solution $\Gamma(\mathcal{T})$ to an instance $\mathcal{T}$ of *AllDNN-Verification* can be partitioned into a collection of rectilinear ϵ-bounded hyperrectangles, $\Gamma(\mathcal{T})$ can be attained by an optimal solution for the *ϵ-RUA-DNN*. This allows as to tackle the *AllDNN-Verification* problem via an efficient approach with strong probabilistic approximation guarantee to solve the *ϵ-RUA-DNN* problem.

Our method is based on two main concepts: the analysis of an underestimated output reachable set with probabilistic guarantees and the *iterative refinement* approach (Wang et al. 2018b). In particular, in Fig. 3 we report a schematic representation of the approach that can be set up through reachable set analysis. Let us consider a possible domain for the safety property, i.e., the polygon highlighted in light blue in the upper left corner of Fig. 3.

Suppose that the undesired output reachable set is the one highlighted in red called $\mathcal{R}^*$ in the bottom left part of the image, i.e., this set describes all the unsafe outcomes the

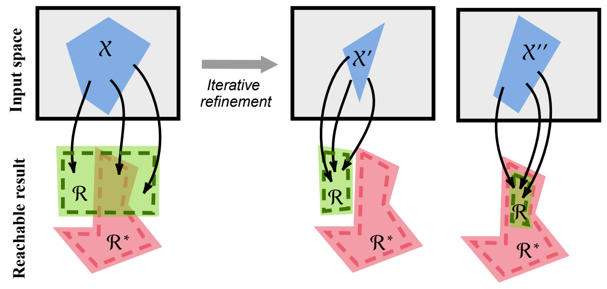

Figure 3: Explanatory image of how to exploit reachable set result for solving the *AllDNN-Verification* problem.

DNN should never output starting from $\mathcal{X}$. Hence, in order to formally verify that the network respects the desired safety property, the output reachable set computed from the domain of the property (i.e., the green $\mathcal{R}$ area of the left side of the image) should have an empty intersection with the undesired reachable set (the red one). If this condition is not respected, as, e.g., in the left part of the figure, then there exists at least an input configuration for which the property is not respected.

To find all the portions of the property's domain where either the undesired reachable set and the output reachable set are disjoint, i.e., $\mathcal{R}^* \bigcap \mathcal{R} = \emptyset$, or, dually, discover the unsafe areas where the condition $\mathcal{R} \subseteq \mathcal{R}^*$ holds (as shown in the right part of Fig. 3) we can exploit the *iterative refinement* approach (Wang et al. 2018b). However, given the nonlinear nature of DNNs, computing the exact output reachable set is infeasible. To address this issue, the reachable set is typically over-approximated, thereby ensuring the soundness of the result. In this vein, (Yang et al. 2022) proposed an enumeration approach based on an over-approximation of the reachable set to compute the set of unsafe regions in the property's input domain. Still, the relaxation of the nonlinear activation functions used to compute the over-approximated reachable set can be computationally demanding. In contrast, we propose a computationally efficient solution that uses underestimation of the reachable set and constructs approximate solutions for the *ϵ-RUA-DNN* problem with strong probabilistic guarantees.

Probabilistic Reachable Set

Given the complexity of computing the exact minimum and maximum of the function computed by a DNN, we propose to approximate the output reachable set using a statistical approach known as *Statistical Prediction of Tolerance Limits* (Wilks 1942; Porter 2019).

We use a Monte Carlo sampling approach: for an appropriately chosen n, we sample n input points and take the smallest and the greatest value achieved in the output node as the lower and the upper extreme of our probabilistic estimate of the reachable set. The choice of the sample size is based on the results of (Wilks 1942) that allow us to choose n in order to achieve a given desired guarantee on the probability α that our estimate of the output reachable set holds

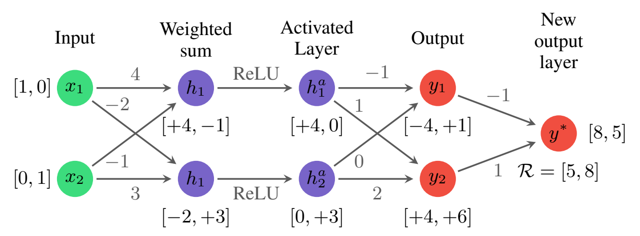

Figure 4: Example of computation single reachable set for a DNN with two outputs.

for at least a fixed (chosen) fraction R of a further possibly infinitely large sample of inputs. Crucially, this statistical result does not require any knowledge of the probability distribution governing our function of interest and thus also applies to general DNNs. Stated in terms more directly applicable to our setting, the main result of (Wilks 1942) is as follows.

Lemma 1. *For any $R \in (0,1)$ and integer n, given a sample of n values from a (continuous) set X the probability that for at least a fraction R of the values in a further possibly infinite sequence of samples from X are all not smaller (respectively larger) than the minimum value (resp. maximum value) estimated with the first n samples is given by the α satisfying the following equation*

$$n \cdot \int_R^1 x^{n-1}\, dx = (1 - R^n) = \alpha \tag{2}$$

Computation of Safe Regions

We are now ready to give a detailed account of our algorithm ϵ-`ProVe`.

Our approximation is based on the analysis of an underestimated output reachable set obtained by sampling a set of n points P_A from a domain of interest A. We start by observing that it is possible to assume, without loss of generality, that the network has a single output node on whose reachable set we can verify the desired property (Liu et al. 2021). For networks not satisfying this assumption, we can enforce it by adding one layer. For example, consider the network in the example of Fig. 4 and suppose we are interested in knowing if, for a given input configuration in a domain $A = [0,1] \times [0,1]$, the output y_1 is always less than y_2. By adding a new output layer with a single node y^* connected to y_1 by weight -1 and to y_2 with weight 1 the condition required reduces to check that all the values in the reachable set for y^* are positive.

In general, from the analysis of the underestimated reachable set of the output node computed as $\mathcal{R} = [min_i\, y_i, max_i\, y_i]$, we can obtain one of these three conditions:

$$\begin{cases} A \text{ is unsafe} & \text{upper bound of } \mathcal{R} < 0 \\ A \text{ is safe} & \text{lower bound of } \mathcal{R} \geq 0 \\ unknown & \text{otherwise} \end{cases} \tag{3}$$

With reference to the toy example in Figure 4, assuming we sample only $n = 2$ input configurations, $(1,0)$ and $(0,1)$

which when propagated through $\mathcal{N}$ produce as a result in the new output layer the vector $y^* = [8, 5]$. This results in the estimated reachable set $\mathcal{R} = [5, 8]$. Since the lower bound of this interval is positive, we conclude that the region A, under consideration, is completely safe.

To confirm the correctness of our construction, we can check the partial values of the original output layer and notice that no input generates $y_1 \geq y_2$. Specifically, if all inputs result in $y_1 \geq y_2$ (violating the specification we are trying to verify), then the reachable set must have, by construction, a negative upper bound, leading to the correct conclusion that the area is unsafe. On the other hand, if only some inputs produce $y_1 \geq y_2$, then we obtain a reachable set with a negative lower bound and a positive upper bound, thus we cannot state whether the area is unsafe or not, and we should proceed with an interval refinement process. Hence, this approach allows us to obtain the situations shown to the right of Fig. 3, i.e., where the reachable set is either completely positive (A safe) or completely negative (A unsafe).

We present the complete pipeline of ϵ−ProVe in Algorithm 1. Our approach receives as input a standard tuple for the *DNN-Verification* and creates the augmented DNN $\mathcal{N}'$ (line 3) following the intuitions provided above. Moreover, we initialize respectively the set of safe regions as an empty set and the unchecked regions as the entire domain of the safety specification encoded in $\mathcal{P}$ (line 5). Inside the loop (line 6), our approximation iteratively considers one area A at the time and begins computing the reachable set, as shown above. We proceed with the analysis of the interval computed, where in case we obtain a positive reachable set, i.e., the lower bound is positive (lines 9-10), then the area under consideration is deemed as safe and stored in the set of safe regions we are enumerating. On the other hand, if the interval is negative, that is, the upper bound is negative, we add the area into the unsafe regions and proceed (lines 11-12). Finally, if we are not in any of these cases, we cannot assert any conclusions about the nature of the region we are checking, and therefore, we must proceed with splitting the area according to the heuristic we prefer (lines 13-14).

The loop ends when either we have checked all areas of the domain of interest or we have reached the ϵ-precision on the iterative refinement. In detail, given the continuous nature of the domain, it is always possible to split an interval into two subparts, that is, the process could continue indefinitely in the worst case. For this reason, as is the case of other state-of-the-art FV methods that are based on this approach, we use a parameter to decide when to stop the process. This does not affect the correctness of the output since our goal is to (tightly) underapproximate the safe regions, and thus, in case the ϵ-precision is reached, the area under consideration would not be considered in the set that the algorithm returns, thus preserving the correctness of the result. Although the level of precision can be set arbitrarily, it does have an effect on the performance of the method. In the supplementary material, we discuss the impact that different heuristics and hyperparameter settings have on the resulting approximation.

Algorithm 1: ϵ−ProVe

```
1: Input: $\mathcal{T} = \langle \mathcal{N}, \mathcal{P}, \mathcal{Q} \rangle$, $n$ (# of samples to compute $\mathcal{R}$), $\epsilon$-precision desired
2: Output: set of safe and unsafe regions in $\mathcal{P}$.
3: $\mathcal{N}' \leftarrow$ CreateAugmentedDNN($\mathcal{N}, \mathcal{P}, \mathcal{Q}$)
4: safe_regions $\leftarrow \emptyset$; unsafe_regions $\leftarrow \emptyset$
5: unknown $\leftarrow$ GetDomain($\mathcal{P}$)
6: while (unknown $\neq \emptyset$) or ($\epsilon$-precision not reached) do
7:     $A \leftarrow$ GetAreaToVerify(unknown)
8:     $\mathcal{R}_A \leftarrow$ ComputeReachableSet($\mathcal{N}'$, $A$, $n$)
9:     if lower($\mathcal{R}_A$) $\geq 0$ then
10:        safe_regions $\leftarrow$ safe_regions $\cup \{A\}$
11:    else if upper($\mathcal{R}_A$) $< 0$ then
12:        unsafe_regions $\leftarrow$ unsafe_regions $\cup \{A\}$
13:    else
14:        unknown $\leftarrow$ unknown $\cup$ IntRefinement($A$)
15:    end if
16: end while
17: return safe_regions, unsafe_regions
```

Theoretical Guarantees

In this section, we analyze the theoretical guarantees that our approach can provide. We assume that the IntRefinement procedure consists of iteratively choosing one of the dimensions of the input domain and splitting the area into two halves of equal size as in (Wang et al. 2018b). The theoretical guarantee easily extends to any other heuristic provided that each split produces two parts both at least a fixed constant fraction β of the subdivided area. Moreover, we assume that reaching the ϵ precision is implemented as testing that the area has reached size ϵ^d,, i.e., it is the cartesian product of d intervals of size ϵ. It follows that, by definition, the areas output by ϵ−ProVe are ϵ-bounded and ϵ-aligned.

The following proposition is the basis of the approximation guarantee (in terms of the size of the safe area returned) on the solution output by ϵ−ProVe on an instance of the ϵ*-RUA-DNN* problem.

Proposition 2. *Fix a real number $\epsilon > 0$, an integer $k \geq 3$, and a real $\gamma > k\epsilon$. Let $\mathcal{T}$ be an instance of the ϵ-RUA-DNN problem. Then for any solution $\mathcal{R} = \{r_1, \ldots, r_m\}$ such that for each $i = 1, \ldots, m$, r_i is γ-bounded, there is a solution $\mathcal{R}^{(\epsilon)} = \{r_1^{(\epsilon)}, \ldots, r_m^{(\epsilon)}\}$ such that each $r_i^{(\epsilon)}$ is ϵ-aligned and $||\mathcal{R}^{(\epsilon)}|| \geq \left(\frac{k-2}{k}\right)^d ||\mathcal{R}||$, where d is the number of dimensions of the input space, and for every solution $\mathcal{R}'$, $||\mathcal{R}'|| = |\cup_i r_i|$ is the total area covered by the hyperrectangles in $\mathcal{R}'$.*

The result is obtained by applying the following lemma to each hyperrectangle of the solution $\mathcal{R}$.

Lemma 3. *Fix a real number $\epsilon > 0$ and an integer $k \geq 3$. For any $\gamma > k\epsilon$ and any γ-bounded rectilinear hyperrectangle $r \subseteq \mathbb{R}^d$, there is an ϵ-aligned rectilinear hyperrectangle $r^{(\epsilon)}$ such that: (i) $r^{(\epsilon)} \subseteq r$; and (ii) $|r^{(\epsilon)}| \geq \left(\frac{k-2}{k}\right)^d |r|$.*

Fig. 5 gives a pictorial explanation of the lemma. In the example shown, $k = 3$ and the parameter ϵ is the unit of

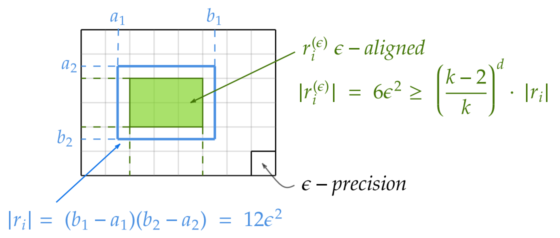
a_1
b_1
a_2
b_2
$r_i^{(\epsilon)}$ ϵ − aligned
$|r_i^{(\epsilon)}| = 6\epsilon^2 \geq \left(\frac{k-2}{k}\right)^d \cdot |r_i|$
ϵ − precision
$|r_i| = (b_1 - a_1)(b_2 - a_2) = 12\epsilon^2$

Figure 5: An example of applying Lemma 3 with $k = 3$.

the grid, which we can imagine superimposed to the bidimensional ($d = 2$) space in which the hyperrectangles live. Hence $\gamma = 3\epsilon$. The non-ϵ-aligned r_i (depicted in blue) is γ-bounded, since its width $w = (b_1 - a_1) = 4\epsilon$ and its height $h = (b_2 - a_2) = 3\epsilon$ are both $\geq 3\epsilon$. Hence, it covers completely at least $w - 2$ columns and $h - 2$ rows of the grid. These rows and columns define the green ϵ-aligned hyperrectangle $r_i^{(\epsilon)}$, of dimension

$$\geq (w-2) \cdot (h-2) \geq \frac{k-2}{k} w \cdot \frac{k-2}{k} h = \left(\frac{k-2}{k}\right)^d |r_i|.$$

The following theorem summarizes the coverage approximation guarantee and the confidence guarantee on the safety nature of the areas returned by ϵ−`ProVe`.

Theorem 4. *Fix a positive integer d and real values $\epsilon, \alpha, R \in (0,1)$, with $R > 1 - \epsilon^d$. Let $\mathcal{T}$ be an instance of the AllDNN-Verification with input in*[2] *$[0,1]^d$, and let k be the largest integer such that $\Gamma(\mathcal{T})$ can be partitioned into $k\epsilon$-bounded rectilinear hyperrectangle.*

Let $\mathcal{R}_+^{(\epsilon)}$ and $\mathcal{R}_-^{(\epsilon)}$ be the sets of areas identified as safe and unsafe, respectively, by ϵ−`ProVe` using n samples at each iteration, with $n \geq \log_R(1 - \alpha^{1/m})$ and $m \geq \max\{|\mathcal{R}_+^{(\epsilon)}|, |\mathcal{R}_-^{(\epsilon)}|\}$. Then, with probability $\geq \alpha$

1. *(coverage guarantee) the solution $\mathcal{R}_+^{(\epsilon)}$ is a $\left(\frac{k-2}{k}\right)^d$ approximation of $\Gamma(\mathcal{T})$, i.e., $||\mathcal{R}_+^{(\epsilon)}|| \geq \left(\frac{k-2}{k}\right)^d |\Gamma(\mathcal{T})|$;*
2. *(safety guarantee) in each hyperrectangle $r \in \mathcal{R}_+^{(\epsilon)}$ at most $(1 - R) \cdot |r|$ points are not safe.*

This theorem gives two types of guarantees on the solution returned by ϵ−`ProVe`. Specifically, point 2. states that for any $R < 1$ and $\alpha < 1$, ϵ−`ProVe` can guarantee that with probability α no more than $(1 - R)$ of the points classified as safe can, in fact, be violations. Moreover, point 1. guarantees that, provided the space of safety points is not too scattered—formalized by the existence of some representation in $k\epsilon$-bounded hyperrectangles— the total area returned by ϵ-ProVe is guaranteed to be close to the actual $\Gamma(\mathcal{T})$.

Finally, the theorem shows that the two guarantees are attainable in an efficient way, providing a quantification of the size n of the sample needed at each iteration. Note that the value of m needed in defining n can be either set using the upper limit $2^{d \log(1/\epsilon)}$— which is the maximum number of possible split operations performed before reaching the ϵ-precision limit—or m can be estimated by a standard doubling technique: repeatedly run the algorithm doubling the estimate for m at each new run until the actual number of areas returned is upper bounded by the current guess for m.

[2]This assumption is w.l.o.g. modulo some normalization.

Proof. The safety guarantee (item 2.) is a direct consequence of Lemma 1. In fact, a hyperrectangle r is returned as safe if all the n sampled points from r are not violations, i.e., their output is ≥ 0 (see (3)). By Lemma 1, at most $(1 - R)$ of the points in r can give an output < 0, with probability $\hat{\alpha} = (1 - R^n)$. Since samples are chosen independently in different hyperrectangles, this bound on the number of violations in a hyperrectangle of $\mathcal{R}^{(\epsilon)}$ holds simultaneously for all of them with probability $\geq \hat{\alpha}^m$. With $n \geq \log_R(1 - \alpha^{1/m})$ we have $\alpha \leq \hat{\alpha}^m$, i.e., the safety guarantee holds with probability $\geq \alpha$.

For the coverage guarantee, we start by noticing that under the hypotheses on k, Proposition 2 guarantees the existence of a solution $\mathcal{R}_1^\epsilon$ made of ϵ-bounded and ϵ-aligned rectilinear hyperrectangles. Let $\mathcal{R}_2^\epsilon$ be a solution obtained from $\mathcal{R}_1^\epsilon$ by partitioning each hyperrectangle into hyperrectangles of minimum possible size ϵ^d, each one ϵ-aligned. The first observation is that, being a solution made of ϵ-bounded and ϵ-aligned rectilinear hyperrectangles, $\mathcal{R}_2^\epsilon$ is among the solutions possibly returned by ϵ−`ProVe`. We now observe that, with probability $\geq \alpha$, each hyperrectangles in $\mathcal{R}_2^\epsilon$ must be contained in some hyperrectangle r in the solution $\mathcal{R}_+^{(\epsilon)}$ returned by ϵ−`ProVe`. First note that if in each iteration of ϵ−`ProVe`, r' keeps on being contained in an area where both safe and violation points are sampled, then eventually r' will become itself an area to analyze. At such a step, clearly every sample in r' will be safe and r' will be included in $\mathcal{R}_+^{(\epsilon)}$, as desired. Therefore, the only possibility for r' not to be contained in any $r \in \mathcal{R}_+^{(\epsilon)}$ is that at some iteration an area $A \supseteq r'$ is analyzed and all the n points sampled in A turn out to be violation points, so A (including r') is classified unsafe (and added to $\mathcal{R}_-^{(\epsilon)}$) by ϵ−`ProVe`. However, by Lemma 1 with probability $(1 - R^n)$ this can happen only if $\epsilon^d = |r'| < (1 - R)|A|$, which contradicts the hypotheses. Hence, with probability $(1 - R^n)^m \geq \alpha$, no hyperrectangle of $\mathcal{R}_-^{(\epsilon)}$ contains r', whence it must be contained in a hyperrectangle of $\mathcal{R}_+^{(\epsilon)}$, concluding the argument. □

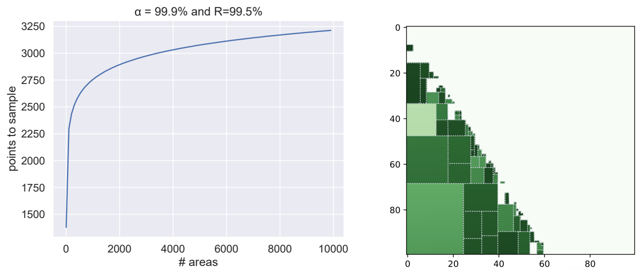

Figure 6: Analysis of ϵ−`ProVe` requirements and results.

Instance	ϵ-**ProVe** ($\alpha = 99.9\%$) # Safe regions	Safe rate	Time	**Exact count** or ***MC sampling*** Safe rate	Time	**Und-estimation (% distance)**
Model_2_20	335	78.50%	0.4s	79.1%	234min	0.74%
Model_2_56	251	43.69%	0.3s	44.46%	196min	1.75%
Model_MN_1	545	64.72%	60.6s	*67.59%*	*0.6s*	4.24%
Model_MN_2	1	100%	0.4s	*100%*	*0.4s*	-
ϕ_2 ACAS Xu_2.1	2462	97.47%	26.9s	99.25%	*0.6s*	1.81%
ϕ_2 ACAS Xu_3.3	1	100%	0.4s	*100%*	*0.5s*	-

Table 1: Comparison of ϵ-ProVe and Exact count or Monte Carlo (MC) sampling approach on different benchmark setups. Full results and other different experiments are reported in the supplementary material.

Fig. 6 (left) shows the correlation between the number of points to be sampled based on the number of areas obtained by ϵ-ProVe if we want to obtain a total confidence of $\alpha = 99.9\%$ and a lower bound $R = 99.5\%$. As we can notice from the plot, if we compute our output reachable set sampling $n = 3250$ points, we are able to obtain the desired confidence and lower bound if the number of regions is in $[1, 10000]$. For this reason, in all our empirical evaluations, we use $n = 3500$ to compute $\mathcal{R}$. An example of the possible result achievable using our approach is depicted in Fig. 6 (right) with different shades of green for better visualization.

Empirical Evaluation

In this section, we evaluate the scalability of our approach, and we validate the theoretical guarantees discussed in the previous section. Our analysis considers both simple DNNs to analyze in detail the theoretical guarantees and two real-world scenarios to evaluate scalability. The first scenario is the ACAS xu (Julian et al. 2016), an airborne collision avoidance system for aircraft, which is a well-known standard benchmark for formal verification of DNNs (Liu et al. 2021; Katz et al. 2017; Wang et al. 2018a). The second scenario considers DNN trained and employed for autonomous mapless navigation tasks in a Deep Reinforcement Learning (DRL) context (Tai, Paolo, and Liu 2017; Marzari, Marchesini, and Farinelli 2023; Marchesini et al. 2023).

All the data are collected on a commercial PC equipped with an M2 Apple silicon. The code used to collect the results and several additional experiments and discussions on the impact of different heuristics for our approximation are available in the supplementary material.

Correctness and Scalability Experiments

These experiments aim to estimate the correctness and scalability of our approach. Specifically for each model tested, we used ϵ-ProVe to return the set of safe regions in the domain of the property under consideration. All data are collected with parameters $\alpha_{TOT} = 99.9\%$ and $R = 99.5\%$ and $n = 3500$ points to compute the reachable set used for the analysis. The results are presented in Table 1. For all experiments, we report the number of safe regions returned by ϵ-ProVe (for which we also know the hyperrectangles position in the property domain), the percentage of safe areas relative to the total starting area (i.e., the safe rate), and the computation time. Moreover, we include a comparison, measured as percentage distance, of the safe rate computed with alternative methods, such as an exact enumeration method (whenever feasible due to the scalability issue discussed above) and a Monte Carlo (MC) Sampling approach using a large number of samples (i.e., 1 million). It's important to note that the MC sampling only provides a probabilistic estimate of the safe rate, lacking information about the location of safe regions in the input domain.

The first block of Table 1 involves two-dimensional models with two hidden layers of 32 nodes activated with ReLU. The safety property consists of all the intervals of $\mathcal{P}$ in the range $[0, 1]$ and a postcondition $\mathcal{Q}$ that encodes a strictly positive output. Notably, ϵ-ProVe is able to return the set of safe regions in a fraction of a second, and the safe rate returned by our approximation deviates at most a 1.75% from the one computed by an exact count, which shows the tightness of the bound returned by our approach. In the second block of Tab. 1, the Mapless Navigation (MN) DNNs are composed of 22 inputs, two hidden layers of 64 nodes activated with $ReLU$, and finally, an output space composed of five nodes, which encode the possible actions of the robot. We test a behavioral safety property where $\mathcal{P}$ encodes a potentially unsafe situation (e.g., *there is an obstacle in front*), and the postcondition $\mathcal{Q}$ specifies the unsafe action that should not be selected. The table illustrates how increasing input space and complexity affects computation time. Nevertheless, the proposed approximation remains efficient even for ACAS xu tests, returning results within seconds. Crucially, focusing on *Model_MN_2* and ϕ_2 ACAS Xu_3.3, ϵ-ProVe states that all the property's domain is safe (i.e., no violation points). The correctness of the results was verified by employing VeriNet (Henriksen et al. 2021), a state-of-the-art FV tool.

Discussion

We studied the *AllDNN-Verification*, a novel problem in the FV of DNNs asking for the set of all the (un)safe regions for a given safety property. Due to the #P-hardness of the problem, we proposed an approximation approach, ϵ-ProVe, which is, to the best of our knowledge, the first method able to efficiently approximate the set of (un)safe regions with some guarantees on the tightness of the solution returned. We believe ϵ-ProVe is an important step to provide consistent and effective tools for analyzing safety in DNNs.

References

Amir, G.; Corsi, D.; Yerushalmi, R.; Marzari, L.; Harel, D.; Farinelli, A.; and Katz, G. 2023. Verifying learning-based robotic navigation systems. In *29th International Conference, TACAS 2023*, 607–627. Springer.

Baluta, T.; Shen, S.; Shinde, S.; Meel, K. S.; and Saxena, P. 2019. Quantitative Verification of Neural Networks and Its Security Applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*.

Bunel, R. R.; Turkaslan, I.; Torr, P.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. Ieee.

Corsi, D.; Marzari, L.; Pore, A.; Farinelli, A.; Casals, A.; Fiorini, P.; and Dall'Alba, D. 2023. Constrained reinforcement learning and formal verification for safe colonoscopy navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Henriksen, P.; et al. 2021. DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis. In *IJCAI*, 2549–2555.

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

Karp, R. M.; and Luby, M. 1985. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *Journal of Complexity*, 1(1): 45–64.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International conference on computer aided verification*, 97–117. Springer.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*.

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

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Marchesini, E.; Marzari, L.; Farinelli, A.; and Amato, C. 2023. Safe Deep Reinforcement Learning by Verifying Task-Level Properties. In *nternational Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Marzari, L.; Corsi, D.; Cicalese, F.; and Farinelli, A. 2023. The #DNN-Verification problem: Counting Unsafe Inputs for Deep Neural Networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

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

Marzari, L.; Marchesini, E.; and Farinelli, A. 2023. Online Safety Property Collection and Refinement for Safe Deep Reinforcement Learning in Mapless Navigation. In *International Conference on Robotics and Automation (ICRA)*.

O'Shea, K.; and Nash, R. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Porter, N. 2019. Wilks' formula applied to computational tools: A practical discussion and verification. *Annals of Nuclear Energy*, 133: 129–137.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

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

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018a. Efficient formal safety analysis of neural networks. *Advances in Neural Information Processing Systems*, 31.

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

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 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.

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

Yang, X.; Yamaguchi, T.; Tran, H.-D.; Hoxha, B.; Johnson, T. T.; and Prokhorov, D. 2022. Neural network repair with reachability analysis. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 221–236. Springer.

Zhang, H.; Wang, S.; Xu, K.; Li, L.; Li, B.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2022a. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems*, 35: 1656–1670.

Zhang, H.; Wang, S.; Xu, K.; Wang, Y.; Jana, S.; Hsieh, C.-J.; and Kolter, Z. 2022b. A branch and bound framework for stronger adversarial attacks of relu networks. In *International Conference on Machine Learning*, 26591–26604. PMLR.